Group 45 Final Report

Rohit George, Yolanda Li, Amanpreet Puri, Rohan Aluri, Vincent Wu

Gantt Chart:
https://docs.google.com/spreadsheets/d/1koO1qefJ67JuQ5YwTL37DSiOR3pZXrR947pKGPtXDlY/edit?usp=drive_link

Contribution Table for Final Report:

Name	Final Report Contributions
Yolanda Li	Video Presentation
Rohit George	Machine Learning Model #2 Implementation
Amanpreet Puri	Machine Learning Model #3 Implementation
Rohan Aluri	Analysis of Models
Vincent Wu	Visualizations for ML Models

Section 1: Introduction/Background

Several studies have experimented with using machine learning to make more accurate NBA predictions regarding sports betting. [2] uses a probabilistic model with data on individual player's history to calculate expected points per possession. However, through further analysis, we concluded that most studies use team-level statistics. [3] was able to achieve an overall 70% accuracy with linear regression, logistic regression, support vector machines, and artificial neural networks using team-level data, with models that combined predictions from multiple sources producing the best results. [1] combined team-level statistics with clustering analysis of players based on playing style to achieve over 70% prediction accuracy. Consequently, based on these results, we aimed to refine our model from the project midterm to focus on predicting the point total for individual NBA games. Oftentimes, users can choose to bet over or under the projected point total. As this value is binary, we believed that analyzing historical team-data would produce more consistent results, thus allowing our model to have greater accuracy. Some of the different factors we have to take into account for this model include: home versus away performance, head-to-head matchups, win-loss records, and player injuries. We intend to constrict our data to the most recent 2-3 years to accommodate roster modifications and performance trends with the current coaching and front office personnel.

However, the most robust preprocessed data we found was data which was from the 2012-2021 NBA seasons. Since there are many features being calculated, we wanted to account for this by including more sample data for the SVM we run in our third supervised algorithm. This amounts to about 7600 NBA games spanning this period. This dataset was processed to include information available for a next-game forecast, meaning the previous 7 day rolling averages were calculated for each team on each date in this time span. Because some of these features are more correlated, an SVM model is a good choice because it can handle these linear and non-linear irregularities.

Information Regarding Datasets:

- General Overview of Home vs Away Data
    - Contains information for all 30 teams using a Team ID, as well as a individual Game ID for every game each team experienced between the 2004 season to now
    - Provides team-level statistics such as number of points scored at home/away, free throw percentage at home/away games, as well as number of assists and rebounds
    - Includes advanced individual player statistics including points per game, rebounds per game, shot percentage, and number of assists for players on each team
- Individual Player Data
    - Contains data including height, weight, and college of players between 1996-2021 season
    - Additional information such as what pick the player was drafted as and their country of origin are included as well
- Team Records
    - Provides a record of all 30 team's winning percentage and record throughout each season
    - Listed per season outcome in chronological order, and organized alphabetically by team
- NBA Team Level Statistics
    - Using field goal percentage, 3-point percentage, free throw percentage, number of offensive/defensive rebounds, assists, steals, turnover, and blocks, this dataset ranks the NBA teams based off their performance in such categories
    - Provides a wide range of statistics depending on conditions such as: per game stats, total stats, per 100 possessions stats, shooting stats, and advanced stats
    - Includes the conference and division standings per season as well as award winners

Section 2: Problem and Motivation

The amount of uncertainty involved in NBA games makes betting outcomes difficult to predict. We noted that individual player betting propositions are influenced by many factors: performance, subtle injuries, location, psychological state, etc., which are often hard to reliably predict as there is very limited public data. Additionally, less implicit features, such as back-to-back games, travel distance, and specific matchups all impact player performance which can produce inconsistent results over time. Additionally, it is also important to note that sports betting applications are highly volatile. This means that odds change quickly throughout the day as the application adjusts to incorporate new information. This leaves very limited opportunities to place a high value bet on an individual player. Due to these circumstances, our group decided to redirect our focus to team-level betting, as there are less variables to account for which can produce more accurate results.

Our primary motivation is to enhance the accuracy of NBA betting predictions. Our project is oriented on determining whether we will go over or under the projected point total (which we will refer to as the "over-under"). Through utilizing historically available data and machine learning models, we aim to predict individual game outcomes with relatively high accuracy. In return, individuals may potentially witness financial rewards. In cases where our machine learning model anticipates that the less favored outcome will occur, then the payout for betting on the underdog line will increase. Since determining "over" or "under" is a binary value, users will also have a heightened chance of witnessing financial returns. Several implications of sports betting include increased thrill and engagement when watching a game. Betting over/under is relatively straightforward and adds an extra layer of interest for users.

Section 3: Methods

Data Preprocessing Methods:

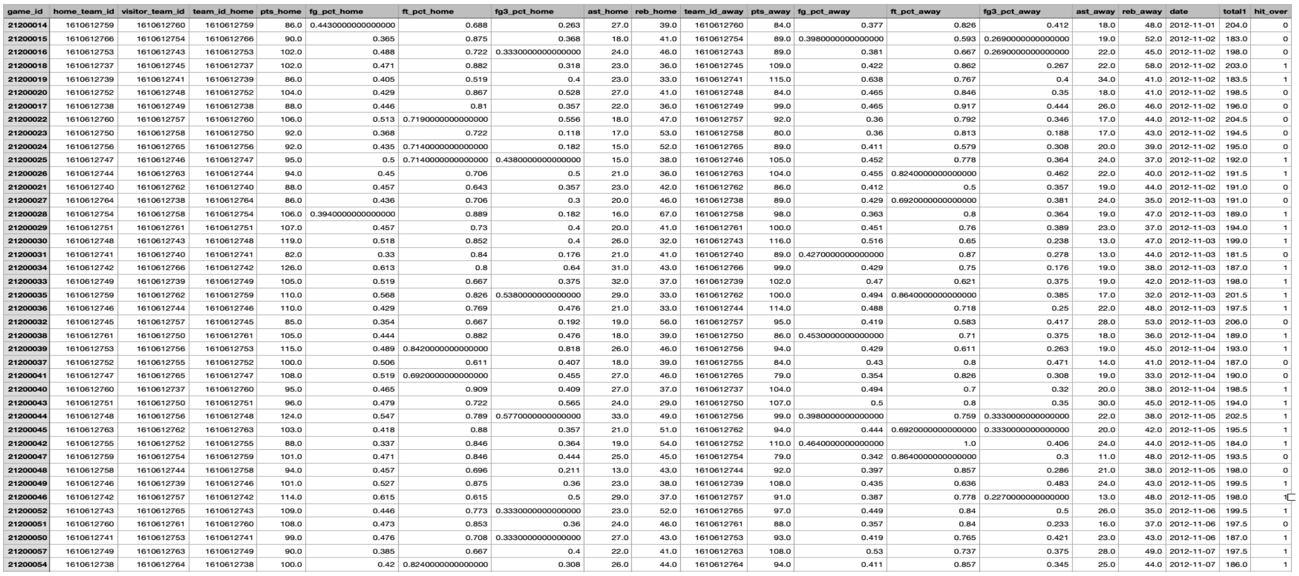
The first dataset we looked at contained NBA statistics for each season. They were individual tables, covering data from 2020-2023. We had to clean and alter these tables in a few ways. The first thing we did was to remove the unnecessary columns, such as games played and minutes played (as each team played a nearly equivalent number of games and minutes when we did not consider playoff games). We then removed the row that covered the sum of stats across all teams in the NBA, as we want to differentiate between teams, not treat them as one group. We then renamed a few columns that had confusing titles. We did this for each of the four tables. Finally, we combined these tables and sorted them by team. This means that every four rows covered a different team, with each row in each group of four covering the data for a team for the respective year. As such, we would have important seasonal statistics for each team over a course of four years, providing multiple features and observations. A lot of this was done by hand, rather than by code.

The next dataset we used was more complicated and much larger. It contained two important tables: the first contained data for each game played in the NBA from 2004-2020, which is understandably a very large number of games. It divided this data into home and away teams, along with the general performance for the home team vs away team. This provided important data on how each team performs when playing at home vs playing away from home. However, there were a few problems with this dataset. The first was that rather than having the team name, it utilized the NBA team ID, which was much more difficult to work with. This brings us to the second table. The second table mapped each NBA team ID to the team nickname. As such, we decided to merge the two tables based on their team ID. Before doing this, we deleted unnecessary columns in this second table, only keeping the team ID and nickname.We also decided to divide the table into two dataframes, with one containing home data for each team and the other containing away. This meant that for the first dataframe, we merged based off Away Team ID in the first table to the NBA team ID in the second table, giving us the nickname of each away team and allowing us to analyze how each team plays away from home.

For the second dataframe, we merged based off Home Team ID to the NBA team ID, giving us the nickname of the home teams and allowing us to analyze how each team performs at home. Overall, both tables kept home and away game data (such as 2 point%, total points, etc.) so we could analyze both how many points a team scored, but also how many they allowed. We also sorted both tables based on their respective nicknames. Since there was too much data, we also added the GAME_DATE_EST column back and cut down on all games before 2018. This both lessened the data our model needed to go through, while also removing data that is highly irrelevant to current team performances. We repeated this for 2012-2021 too in order to increase our preprocessing data and use 2022 for testing purposes.

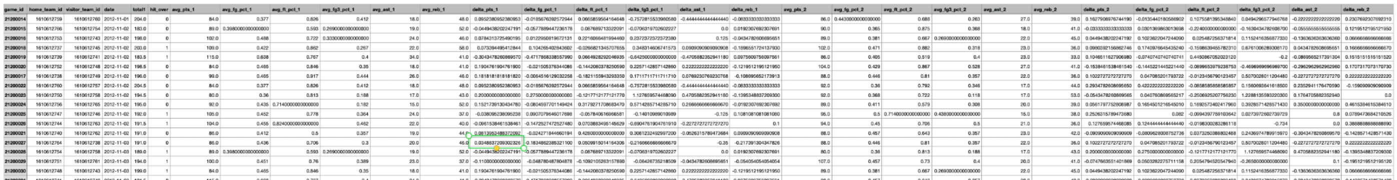
We then preprocessed the data again so now for each game there are rolling averages computed for each of the main statistics as well as the percent change over the last 7 days. Our newly processed datafiles can be accessed through Github.

Before Data Processing: Original dataset box score recording every game for given time period



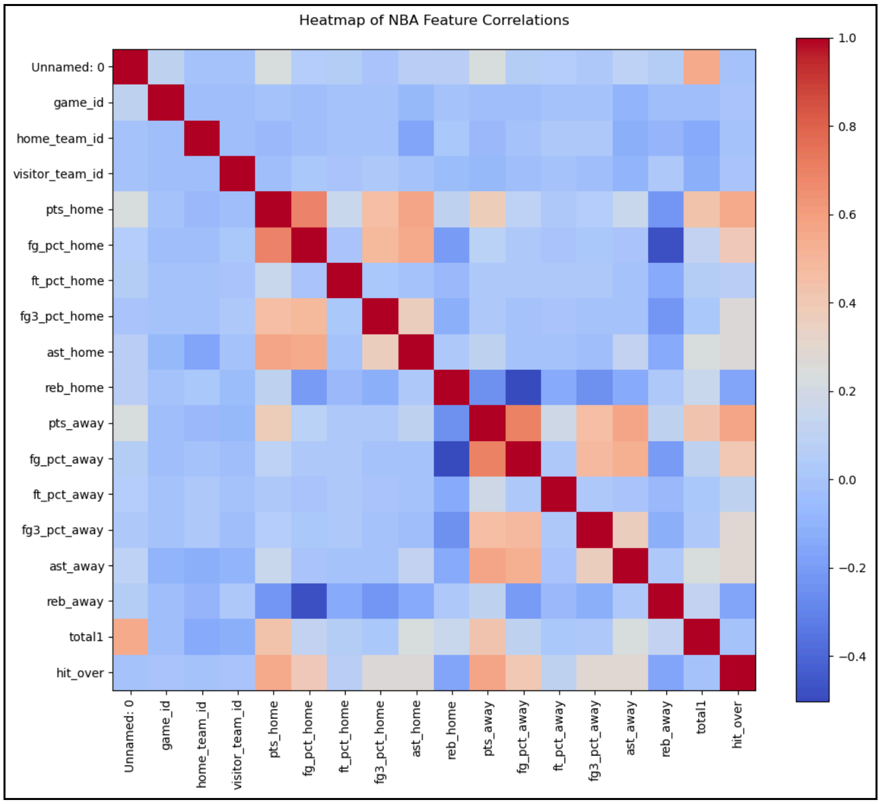
game_id	home_team_id	visitor_team_id	team_id_home	pts_home	fg_pct_home	ft_pct_home	fg3_pct_home	ast_home	reb_home	team_id_away	pts_away	fg_pct_away	ft_pct_away	fg3_pct_away	ast_away	reb_away	date	total1	hit_over
21200014	1610612759	1610612760	1610612759	86.0	0.4430000000000000	0.688	0.263	27.0	39.0	1610612760	84.0	0.377	0.826	0.412	18.0	48.0	2012-11-01	204.0	0
21200015	1610612766	1610612754	1610612766	90.0	0.365	0.875	0.368	18.0	41.0	1610612754	89.0	0.3980000000000000	0.593	0.2690000000000000	19.0	52.0	2012-11-02	183.0	0
21200016	1610612753	1610612743	1610612753	102.0	0.488	0.722	0.3330000000000000	24.0	46.0	1610612743	89.0	0.381	0.667	0.2690000000000000	22.0	45.0	2012-11-02	198.0	0
21200018	1610612737	1610612745	1610612737	102.0	0.471	0.882	0.318	23.0	36.0	1610612745	109.0	0.422	0.862	0.267	22.0	58.0	2012-11-02	203.0	1
21200019	1610612739	1610612741	1610612739	86.0	0.405	0.519	0.4	23.0	33.0	1610612741	115.0	0.638	0.767	0.4	34.0	41.0	2012-11-02	183.5	1
21200020	1610612752	1610612748	1610612752	104.0	0.429	0.867	0.528	27.0	41.0	1610612748	84.0	0.465	0.846	0.35	18.0	41.0	2012-11-02	198.5	0
21200017	1610612738	1610612749	1610612738	88.0	0.446	0.81	0.357	22.0	36.0	1610612749	99.0	0.465	0.917	0.444	26.0	46.0	2012-11-02	196.0	0
21200022	1610612760	1610612757	1610612760	106.0	0.513	0.7190000000000000	0.556	18.0	47.0	1610612757	92.0	0.36	0.792	0.346	17.0	44.0	2012-11-02	204.5	0
21200023	1610612750	1610612758	1610612750	92.0	0.368	0.722	0.118	17.0	53.0	1610612758	80.0	0.36	0.813	0.188	17.0	43.0	2012-11-02	194.5	0
21200024	1610612756	1610612765	1610612756	92.0	0.435	0.7140000000000000	0.182	15.0	52.0	1610612765	89.0	0.411	0.579	0.308	20.0	39.0	2012-11-02	195.0	0
21200025	1610612747	1610612746	1610612747	95.0	0.5	0.7140000000000000	0.4380000000000000	15.0	38.0	1610612746	105.0	0.452	0.778	0.364	24.0	37.0	2012-11-02	192.0	1
21200026	1610612744	1610612763	1610612744	94.0	0.45	0.706	0.5	21.0	36.0	1610612763	104.0	0.455	0.8240000000000000	0.462	22.0	40.0	2012-11-02	191.5	1
21200021	1610612740	1610612762	1610612740	88.0	0.457	0.643	0.357	23.0	42.0	1610612762	86.0	0.412	0.5	0.357	19.0	44.0	2012-11-02	191.0	0
21200027	1610612764	1610612738	1610612764	86.0	0.436	0.706	0.3	20.0	46.0	1610612738	89.0	0.429	0.6920000000000000	0.381	24.0	35.0	2012-11-03	191.0	0
21200028	1610612754	1610612758	1610612754	106.0	0.3940000000000000	0.889	0.182	16.0	67.0	1610612758	98.0	0.363	0.8	0.364	19.0	47.0	2012-11-03	189.0	1
21200029	1610612751	1610612761	1610612751	107.0	0.457	0.73	0.4	20.0	41.0	1610612761	100.0	0.451	0.76	0.389	23.0	37.0	2012-11-03	194.0	1
21200030	1610612748	1610612743	1610612748	119.0	0.518	0.852	0.4	26.0	32.0	1610612743	116.0	0.516	0.65	0.238	13.0	47.0	2012-11-03	199.0	1
21200031	1610612741	1610612740	1610612741	82.0	0.33	0.84	0.176	21.0	41.0	1610612740	89.0	0.4270000000000000	0.87	0.278	13.0	44.0	2012-11-03	181.5	0
21200034	1610612742	1610612766	1610612742	126.0	0.613	0.8	0.64	31.0	43.0	1610612766	99.0	0.429	0.75	0.176	19.0	38.0	2012-11-03	187.0	1
21200033	1610612749	1610612739	1610612749	105.0	0.519	0.667	0.375	32.0	37.0	1610612739	102.0	0.47	0.621	0.375	19.0	42.0	2012-11-03	198.0	1
21200035	1610612759	1610612762	1610612759	110.0	0.568	0.826	0.5380000000000000	29.0	33.0	1610612762	100.0	0.494	0.8640000000000000	0.385	17.0	32.0	2012-11-03	201.5	1
21200036	1610612746	1610612744	1610612746	110.0	0.429	0.769	0.476	21.0	33.0	1610612744	114.0	0.488	0.718	0.25	22.0	48.0	2012-11-03	197.5	1
21200032	1610612745	1610612757	1610612745	85.0	0.354	0.667	0.192	19.0	56.0	1610612757	95.0	0.419	0.583	0.417	28.0	53.0	2012-11-03	206.0	0
21200038	1610612761	1610612750	1610612761	105.0	0.444	0.882	0.476	18.0	39.0	1610612750	86.0	0.4530000000000000	0.71	0.375	18.0	36.0	2012-11-04	189.0	1
21200039	1610612753	1610612756	1610612753	115.0	0.489	0.8420000000000000	0.818	26.0	46.0	1610612756	94.0	0.429	0.611	0.263	19.0	45.0	2012-11-04	193.0	1
21200037	1610612752	1610612755	1610612752	100.0	0.506	0.611	0.407	18.0	39.0	1610612755	84.0	0.43	0.8	0.471	14.0	41.0	2012-11-04	187.0	0
21200041	1610612747	1610612765	1610612747	108.0	0.519	0.6920000000000000	0.455	27.0	46.0	1610612765	79.0	0.354	0.826	0.308	19.0	33.0	2012-11-04	190.0	0
21200040	1610612760	1610612737	1610612760	95.0	0.465	0.909	0.409	27.0	37.0	1610612737	104.0	0.494	0.7	0.32	20.0	38.0	2012-11-04	198.5	1
21200043	1610612751	1610612750	1610612751	96.0	0.479	0.722	0.565	24.0	29.0	1610612750	107.0	0.5	0.8	0.35	30.0	45.0	2012-11-05	194.0	1
21200044	1610612748	1610612756	1610612748	124.0	0.547	0.789	0.5770000000000000	33.0	49.0	1610612756	99.0	0.3980000000000000	0.759	0.3330000000000000	22.0	38.0	2012-11-05	202.5	1
21200045	1610612763	1610612762	1610612763	103.0	0.418	0.88	0.357	21.0	51.0	1610612762	94.0	0.444	0.6920000000000000	0.3330000000000000	20.0	42.0	2012-11-05	195.5	1
21200042	1610612755	1610612752	1610612755	88.0	0.337	0.846	0.364	19.0	54.0	1610612752	110.0	0.4640000000000000	1.0	0.406	24.0	44.0	2012-11-05	184.0	1
21200047	1610612759	1610612754	1610612759	101.0	0.471	0.846	0.444	25.0	45.0	1610612754	79.0	0.342	0.8640000000000000	0.3	11.0	48.0	2012-11-05	193.5	0
21200048	1610612758	1610612744	1610612758	94.0	0.457	0.696	0.211	13.0	43.0	1610612744	92.0	0.397	0.857	0.286	21.0	38.0	2012-11-05	198.0	0
21200049	1610612746	1610612739	1610612746	101.0	0.527	0.875	0.36	23.0	38.0	1610612739	108.0	0.435	0.636	0.483	24.0	43.0	2012-11-05	199.5	1
21200046	1610612742	1610612757	1610612742	114.0	0.615	0.615	0.5	29.0	37.0	1610612757	91.0	0.387	0.778	0.2270000000000000	13.0	48.0	2012-11-05	198.0	1
21200052	1610612743	1610612765	1610612743	109.0	0.446	0.773	0.3330000000000000	23.0	52.0	1610612765	97.0	0.449	0.84	0.5	26.0	35.0	2012-11-06	199.5	1
21200051	1610612760	1610612761	1610612760	108.0	0.473	0.853	0.36	24.0	46.0	1610612761	88.0	0.357	0.84	0.233	16.0	37.0	2012-11-06	197.5	0
21200050	1610612741	1610612753	1610612741	99.0	0.476	0.708	0.3330000000000000	27.0	43.0	1610612753	93.0	0.419	0.765	0.421	23.0	43.0	2012-11-06	187.0	1
21200057	1610612749	1610612763	1610612749	90.0	0.385	0.667	0.4	22.0	41.0	1610612763	108.0	0.53	0.737	0.375	28.0	49.0	2012-11-07	197.5	1
21200054	1610612738	1610612764	1610612738	100.0	0.42	0.8240000000000000	0.308	26.0	44.0	1610612764	94.0	0.411	0.857	0.345	25.0	44.0	2012-11-07	186.0	1

After Data Processing: Processed dataset involving rolling 7-day averages and rate of change of each statistic for each team in the matchup



game_id	home_team_id	visitor_team_id	date	total1	hit_over	avg_pts_1	avg_fg_pct_1	avg_ft_pct_1	avg_fg3_pct_1	avg_ast_1	avg_reb_1	delta_pts_1	delta_fg_pct_1	delta_ft_pct_1	delta_fg3_pct_1	delta_ast_1	delta_reb_1	avg_pts_2	avg_fg_pct_2	avg_ft_pct_2	avg_fg3_pct_2	avg_ast_2	avg_reb_2	delta_pts_2	delta_fg_pct_2	delta_ft_pct_2	delta_fg3_pct_2	delta_ast_2	delta_reb_2
21200014	1610612759	1610612760	2012-11-01	204.0	0	84.0	0.377	0.826	0.412	18.0	48.0	0.0952380952380953	-0.0185676392572944	0.0665859564164648	-0.7572815533980580	-0.4444444444444440	-0.0833333333333333	86.0	0.4430000000000000	0.688	0.263	27.0	39.0	0.1627906976744190	-0.0135440180586902	0.1075581395348840	0.0494296577946768	-0.2222222222222220	0.2307692307692310
21200015	1610612766	1610612754	2012-11-02	183.0	0	89.0	0.3980000000000000	0.593	0.2690000000000000	19.0	52.0	-0.0449438202247191	-0.0577889447236178	0.087689713322091	-0.0706319702602227	0.0	0.0192307692307691	90.0	0.365	0.875	0.368	18.0	41.0	-0.0333333333333333	0.0301369863013698	-0.2240000000000000	-0.1630434782608700	-0.0555555555555555	0.1219512195121950
21200016	1610612753	1610612743	2012-11-02	198.0	0	102.0	0.488	0.722	0.3330000000000000	24.0	46.0	0.0784313725490195	0.0122950819672131	0.2216066481994460	0.2372372372372380	0.125	-0.0434782608695651	89.0	0.381	0.667	0.2690000000000000	22.0	45.0	0.0449438202247192	0.1023622047244090	0.025487256371814	0.1152416356877330	-0.1363636363636360	0.0666666666666666
21200018	1610612737	1610612745	2012-11-02	203.0	1	109.0	0.422	0.862	0.267	22.0	58.0	0.073394495412844	0.104265402843602	-0.0266821345707655	0.348314606741573	0.0909090909090908	-0.1896551724137930	102.0	0.471	0.882	0.318	23.0	36.0	0.0980392156862746	0.1740976645435240	-0.1598639455782310	0.6761006289308170	0.0434782608695651	0.1666666666666670
21200019	1610612739	1610612741	2012-11-02	183.5	1	115.0	0.638	0.767	0.4	34.0	41.0	-0.3043478260869570	-0.4717868338557990	0.0664928292046935	-0.6425000000000000	-0.4705882352941180	0.0975609756097561	86.0	0.405	0.519	0.4	23.0	33.0	0.1046511627906980	-0.0740740740740741	0.4450867052023120	-0.2	-0.0869565217391304	0.1515151515151520
21200020	1610612752	1610612748	2012-11-02	198.5	0	84.0	0.465	0.846	0.35	18.0	41.0	0.1904761904761900	-0.021505376344086	-0.1442080378250590	0.2257142857142860	0.2222222222222220	-0.1219512195121950	104.0	0.429	0.867	0.528	27.0	41.0	-0.1538461538461540	-0.1445221445221440	-0.0899653979238753	-0.4696969696969700	-0.2962962962962960	0.1707317073170730
21200017	1610612738	1610612749	2012-11-02	196.0	0	99.0	0.465	0.917	0.444	26.0	46.0	0.1818181818181820	-0.0064516129032258	-0.1821155943293350	0.1711711711711710	0.0769230769230768	-0.108695652173913	88.0	0.446	0.81	0.357	22.0	36.0	0.1022727272727270	0.047085201793722	-0.0123456790123457	0.5070028011204480	-0.2272727272727270	0.2222222222222220
21200022	1610612760	1610612757	2012-11-02	204.5	0	84.0	0.377	0.826	0.412	18.0	48.0	0.0952380952380953	-0.0185676392572944	0.0665859564164648	-0.7572815533980580	-0.4444444444444440	-0.0833333333333333	92.0	0.36	0.792	0.346	17.0	44.0	0.2934782608695650	0.4222222222222220	-0.0858585858585857	0.1560693641618500	0.2352941176470590	-0.159090909090909
21200023	1610612750	1610612758	2012-11-02	194.5	0	80.0	0.36	0.813	0.188	17.0	43.0	0.2000000000000000	0.2750000000000000	-0.1217712177121770	1.127659574468090	0.4705882352941180	-0.1395348837209300	92.0	0.368	0.722	0.118	17.0	53.0	-0.0543478260869565	0.040760869565217	-0.2506925207756230	1.2288135593220300	0.1764705882352940	0.0
21200024	1610612756	1610612765	2012-11-02	195.0	0	92.0	0.435	0.7140000000000000	0.182	15.0	52.0	0.1521739130434780	-0.0804597701149424	0.3179271708683470	0.5714285714285710	0.2666666666666670	-0.0192307692307692	89.0	0.411	0.579	0.308	20.0	39.0	0.0561797752808987	0.1654501216545010	0.1692573402417960	0.3928571428571430	0.3500000000000000	0.4615384615384610
21200025	1610612747	1610612746	2012-11-02	192.0	1	105.0	0.452	0.778	0.364	24.0	37.0	-0.038095238095238	0.0907079646017698	-0.057840616966581	-0.14010989010989	-0.125	0.1081081081081080	95.0	0.5	0.7140000000000000	0.4380000000000000	15.0	38.0	0.2526315789473680	0.082	-0.0994397759103642	0.0273972602739729	0.8	0.0789473684210526
21200026	1610612744	1610612763	2012-11-02	191.5	1	104.0	0.455	0.8240000000000000	0.462	22.0	40.0	-0.0961538461538461	-0.1472527472527480	0.0703883495145629	-0.6904761904761910	-0.2272727272727270	0.1	94.0	0.45	0.706	0.5	21.0	36.0	0.127659574468085	0.1244444444444440	-0.0198300283286118	-0.734	0.0	0.3888888888888890
21200021	1610612740	1610612762	2012-11-02	191.0	0	86.0	0.412	0.5	0.357	19.0	44.0	0.0813953488372092	-0.024271844660194	0.4280000000000000	0.3081232492997200	-0.0526315789473684	0.0909090909090908	88.0	0.457	0.643	0.357	23.0	42.0	-0.0909090909090909	-0.0809628008752736	0.0373250388802488	0.2436974789915970	-0.3043478260869570	-0.1428571428571430
21200027	1610612764	1610612738	2012-11-03	191.0	0	86.0	0.436	0.706	0.3	20.0	46.0	0.0348837209302326	-0.1834862385321100	0.0509915014164306	-0.2166666666666670	-0.35	-0.217391304347826	88.0	0.446	0.81	0.357	22.0	36.0	0.1022727272727270	0.047085201793722	-0.0123456790123457	0.5070028011204480	-0.2272727272727270	0.2222222222222220
21200028	1610612754	1610612758	2012-11-03	189.0	1	89.0	0.3980000000000000	0.593	0.2690000000000000	19.0	52.0	-0.0449438202247191	-0.0577889447236178	0.087689713322091	-0.0706319702602227	0.0	0.0192307692307691	80.0	0.36	0.813	0.188	17.0	43.0	0.2000000000000000	0.2750000000000000	-0.1217712177121770	1.127659574468090	0.4705882352941180	-0.1395348837209300

Heatmap to Determine NBA Feature Correlations:



We decided to implement a heatmap to determine the level of correlation between all possible combinations of variables within our dataset. This visualization helps users understand which variables are positively or negatively related to one another depending on their color according to the legend on the right. Warm colors in this case represent positive correlations, while cooler colors represent negative correlation. This helps our group significantly as we can come to deductions such as points at home are strongly and positively correlated with assists at home (the color appears orange), which helps us conclude that if an NBA team has a high number of assists, then it is very likely that they have a high number of points as well. Through the heatmap, we can have insights regarding which variables we potentially don't need to include within our model whether it be because it contributes similar information to another variable or because it is irrelevant and provides unnecessary information.

ML Algorithm #1: Logistic Regression

- Code for Logistic Regression Supervised Learning Method: Available on Github
- Summary Of Process for Creating Data Model: We used logistic regression to predict if we can forecast the next game going over the betting over-under total on points. To do this, the features mentioned previously, field goal percentage (FG%), 3 point percentage (3P%), free throw percentage (FT%), offensive rebounds (ORB), defensive rebounds (DRB), assists (AST), steals (STL), etc. are fed into a logistic regression algorithm. We chose to use a logistic regression algorithm because it allows us to perform binary classification. "Over" and "under" are the two possible results we can have, so binary classification is suitable here. Additionally, logistic regression provides easily interpretable results; the coefficients represent the impact of each feature on the odds of the outcome, making it easy to understand the relationship between over / under outcomes and our independent variables. It is also not as susceptible to overfitting, and

works well with small and large datasets (it is fairly simple and not very computationally expensive). In addition, the logistic regression model can be considered as a supervised learning method as the input-output pairs are correlated. Our file containing the data and results for the logistic regression model can be found on the Github repository.

ML Algorithm #2: Random Forest

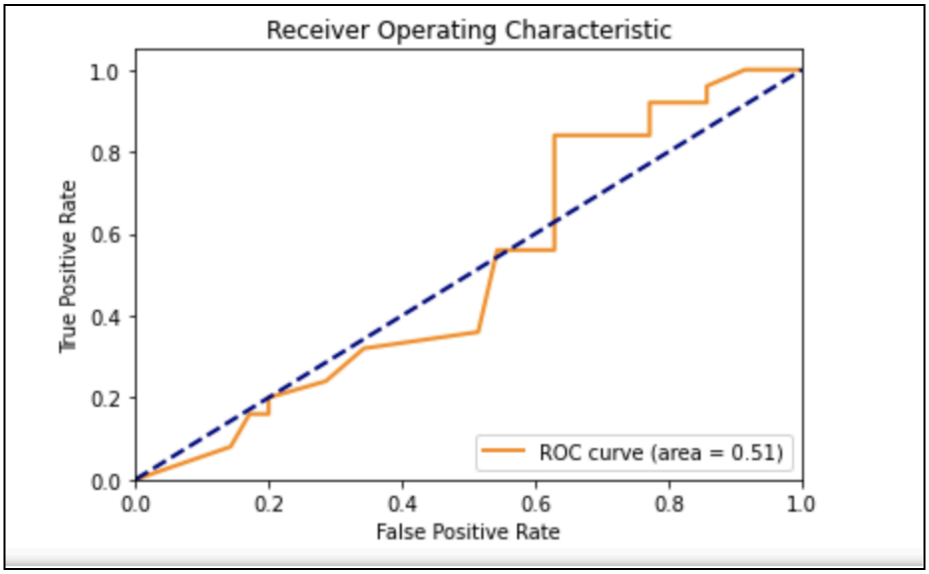
- Code for Random Forest Algorithm: Available on Github
- Summary of Process for Creating Data Model: We chose to do random forest next, mainly due to its ability to handle complex, nonlinear relationships. This is due to the building of multiple decision trees, with each based on a different subset of the data and features. This allows the model to capture complex interactions and dependencies among the features. Once it averages these outputs, we reduce the risk of overfitting and thus reduce the impact of noise. This is important for basketball because it is such a complex sport. It is influenced by so many features, some more important than others. Random Forest can decide which features are most important. To implement random forest, we used the data set of each game with the statistics previously mentioned. We once again had a binary representation of "Over" and "Under," allowing us to use random forest classification to make it the indices of the dataset. We then trained the model on the data, splitting some of it into the testing set, and ran it. Another important note about random forest is that it is quicker to run, due to not needing to scale or normalize the data.

ML Algorithm #3: Support Vector Machine

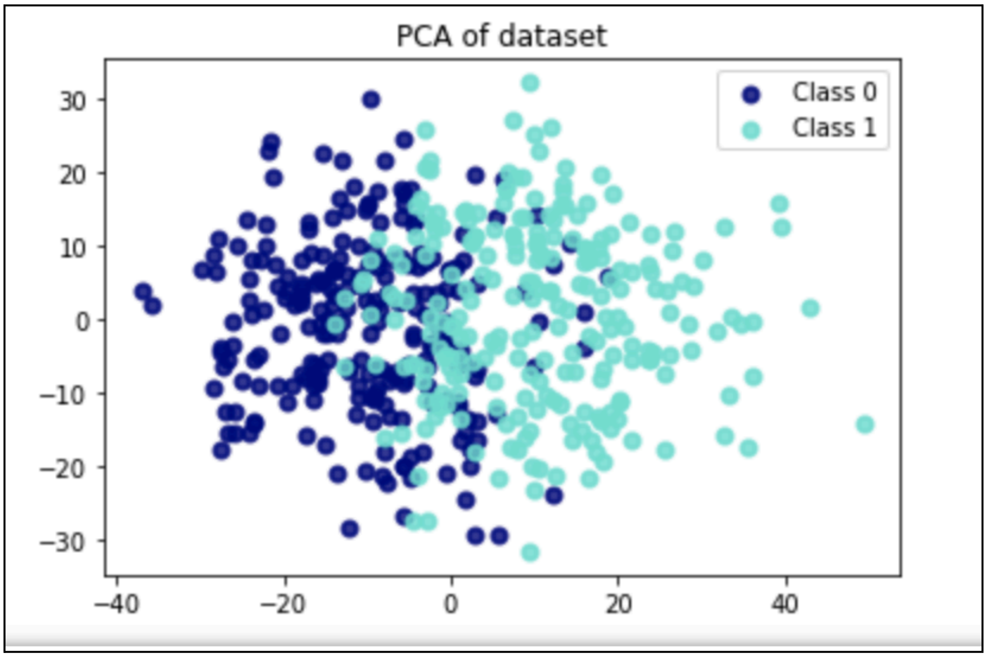
- Code for Support Vector Machine Algorithm: Available on Github
- Summary of Process for Creating Data Model: The SVM model with some tuning was chosen because of its use cases with high dimensional, numerical data. Initially, we were going to use Naive Bayes, but the issue with this approach is Naive Bayes assumes each of the features are independent. In this dataset, that is clearly not the case. Field goal and three-point field goal percentage are in essence overlapping in measurement. The SVM model was chosen as it has a good approach to this by maximizing the hyperplane between these small differences in points. In addition, the SVM is a good choice for binary classification.

Section 4: Results and Discussion

Visualization of Results for Logistic Regression Model:



The first visualization for the logistic regression model consists of a ROC chart which represents the flow of true positive rates versus false positive rates. True positive rate in this case refers to the proportion of “over” cases correctly identified, and false positive rate refers to the proportion of “under” cases incorrectly identified as “over”. We can measure the area under the curve to determine the total accuracy within our implemented model. It appears that the area under the ROC curve is similar to the slope of the reference line (dashed blue line). We can conclude this because our model has a ROC curve area of 0.51, and the random classifier (i.e. can be flipping a coin and landing on heads) represents the probability of true positives and false positives being equal through all thresholds with an ROC of 0.50. Based on the ROC curve, we can determine that there is potential for improvement within our logistic regression model. As we desire a higher value for the area under the ROC curve, we can conclude that there are irrelevant features and noise within the data.



Then, to get additional information regarding how well our model performed, we conducted a PCA component analysis while splitting the data into two distinct components for class analysis. Since our project is oriented on betting over / under, our outcome is a binary variable with one of two options: the combined point total will exceed the over/under, or the combined point total will not exceed the over/under. Therefore we have two classes (Class 0 and Class 1) to represent the allocated outcomes. Immediately, from the visualization, we can tell that there is clear overlap between the plotted points for Class 0 and Class 1. This means that our logistic regression model will potentially struggle to accurately distinguish between accurately betting over the over-under and accurately betting under the over-under as well. It appears there exists a high level of variance within our data as our plotted points appear to be diffused and more spread out.

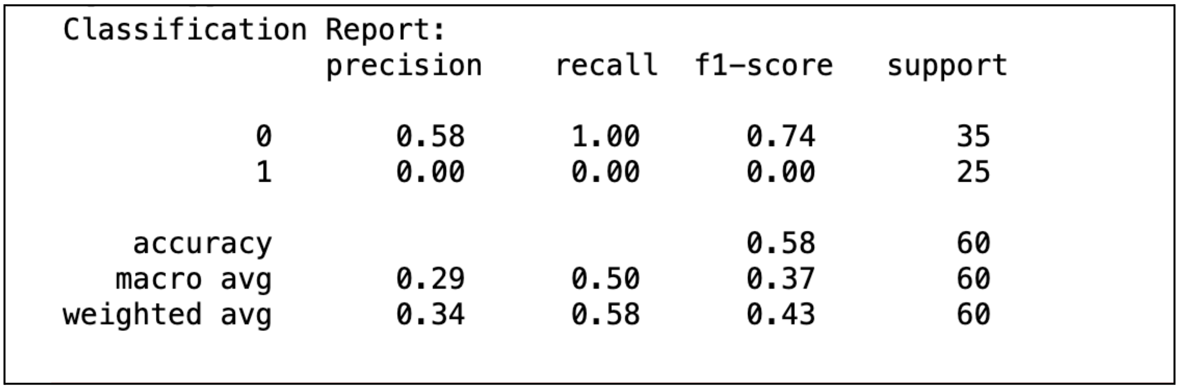
Quantitative Metrics for Logistic Regression Model:

To determine the accuracy of our logistic regression model, we can utilize several quantitative metric techniques. Previously, in the project proposal, we determined that we were going to use balanced accuracy, brier score, and mean squared error as a metric of determining the level of our error. However, since we are only partially done with our implementation, we will use simpler quantitative metrics for now. So far, we have implemented a logistic regression model that categorizes each game as over / under, treated as 1/0 respectively. We can analyze the results of our logistic regression model using precision score, recall score, F1 score, and more, all included in the classification report:

- Precision = # true positives / # predicted positives

- Recall = # true positives / # actual positives

- F1 score = 2 * Precision * Recall / (Precision + Recall)



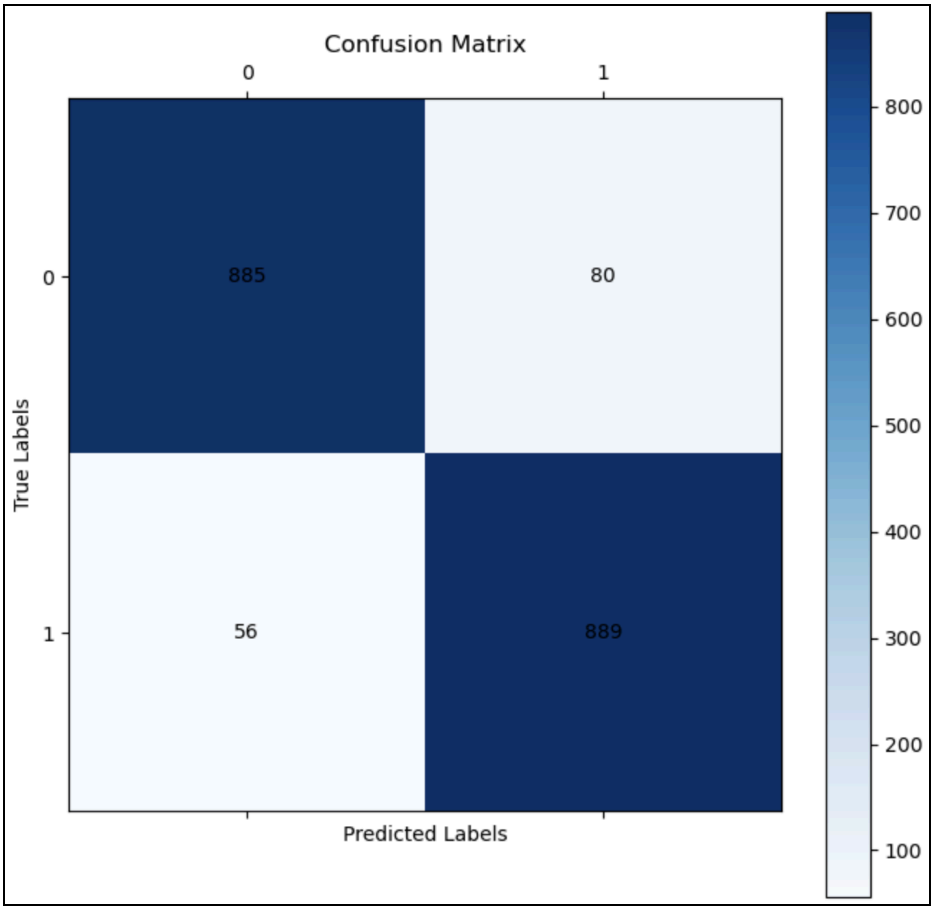
```
Classification Report:
               precision    recall  f1-score   support

           0       0.58      1.00      0.74        35
           1       0.00      0.00      0.00        25

    accuracy                           0.58        60
   macro avg       0.29      0.50      0.37        60
weighted avg       0.34      0.58      0.43        60
```

The screenshot above depicts the report of the three quantitative measures: precision, recall, and f1-score. The accuracy of our data is reported as well, implying that our model is 58% correct. An apparent error within our model is that the value for precision and recall is 0.0 for class 1. This means that an error is occurring when trying to accurately predict when the combined score is greater than the over-under. The quantitative metrics on the whole provide us with further understanding as it explains the numerical performance of our model. Through the values provided, we can conduct multiple trials to train our model and increase the overall efficiency. We decided to implement accuracy and F1-score as our measure of performance because it provides a quick and intuitive way to test our imbalance classes. Through accuracy we can test how correct our model is through all classifications, and through F1-score we can analyze the performance metrics for the positive class.

Analysis of Logistic Regression Model:

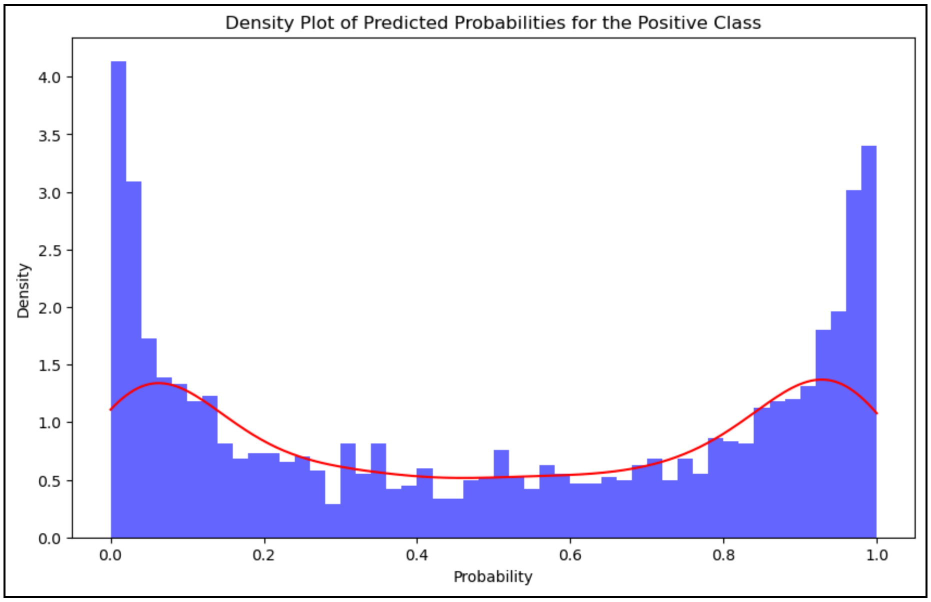
Based on the visualizations and the values obtained in the quantitative metrics subsection, it is evident that our current model has room for improvement. As we are dealing with sports betting applications and monetary value is involved, we want to maximize the accuracy of our model so that user's aren't losing any money. To obtain the results presented in the classification report, we implemented a logistic regression supervised learning method to accurately predict whether a game will go over/under the betting total. Our model consists of various independent variables (FG%, 3P%, FT%, ORB, DRB, AST, STL, etc.) that provide a threshold of projected points. Using these independent variables, we can calculate the dependent variable (how many combined points both teams will score). Additionally, the quantitative metrics allow us to see a representation of the error within our model. Our accuracy rate was 0.58, which is only marginally better than completely random guesses, which would have an expected accuracy of 0.5. We can try to identify the rationale behind why these errors exist. Oftentimes, this may be due to an irregular game where individual players perform poorly arbitrarily, but since the accuracy is low, it may be due to the nature of the model. For instance, logistic regression models are not suitable in conditions where there exists non-linear relationships and noise. In a study conducted by Harvard graduates, a PCA analysis was conducted on a dataset consisting of NBA team level stats over a decade to predict conference ranking for the upcoming season. After conducting the PCA test, only several key basketball statistics (FG%, TV, OR, DR) were highlighted. After making revisions to the model through multiple trials, the final forecast that was conducted by the students attained a mean accuracy score error of 19% indicating there was minimal error. Based on these results, it is apparent there are independent variables influencing our model that are not accurate representations of how well a team may perform. We must identify and remove these excess independent variables to receive more accurate results as desired.

Visualization of Results from Random Forests Model:



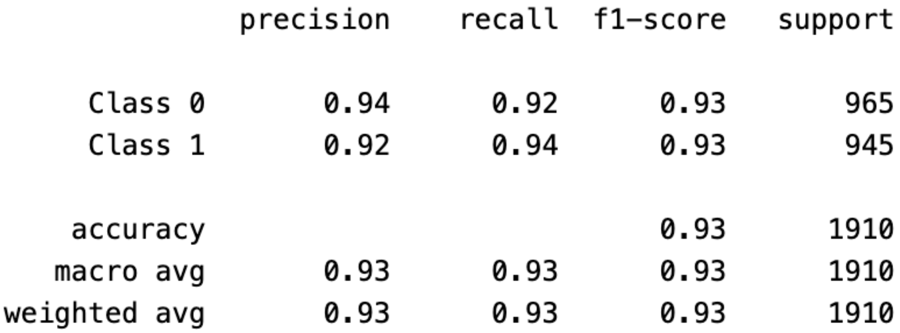
The first visualization for our Random Forests model is a confusion matrix that highlights the values of True Positive, True Negative, False Positive, and False Negative. In this case, True Positive and True Negative refers to the number of correct predictions that the model exceeds the over-under or falls behind the over-under, respectively. False Positive and False Negative refers to the inaccuracies that our model makes. As the top-left hand box represents True Negatives, it is apparent that our model is able to accurately predict when to bet under the over-under for 885 trials which is classified as Class 0. Similarly, the bottom-right hand side represents True Positive, which implies that our model is able to accurately predict when to bet over the over-under for 889 trials which is classified as Class 1. Since we have higher numbers on the diagonal related to the off-diagonal (which we can visually see through a darker color), this indicates that our model has a good predictive performance. The following calculations can be implemented for further analysis simply based off of the provided confusion matrix:

- Accuracy: (TP + TN) / Total = (889 + 885) / (885 + 80 + 56 + 889) = 0.9288
- Precision (Positive Predicted Value): TP / (TP + FP) = 889 / (889 + 80) = 0.9174
- Recall (True Positive Rate): TP / (TP + FN) = 889 / (889 + 56) = 0.9407



Our second visualization for the Random Forest model is a density plot of predicted probabilities for the positive class. The red line represents a sample fitted distribution for how the density of the predicted probabilities is expected to look. We can notice that there are evident peaks in the red line when the probability is closest to 0.0 and 1.0. Based on the histogram in blue, we see that our model produced values similar to the sample red line as it follows bimodal distribution where peaks are located on the boundaries at 0.0 and 1.0 as well. Since our histogram has peaks on opposite ends and is relatively flat in the middle, this implies that the model is making very confident predictions regarding whether the projected point total is accurately over or under the over-under. Values in the 1.0 column represent when the model accurately predicts the over, and vice versa for the values in the 0.0 column.

Quantitative Metrics for Random Forest Model:



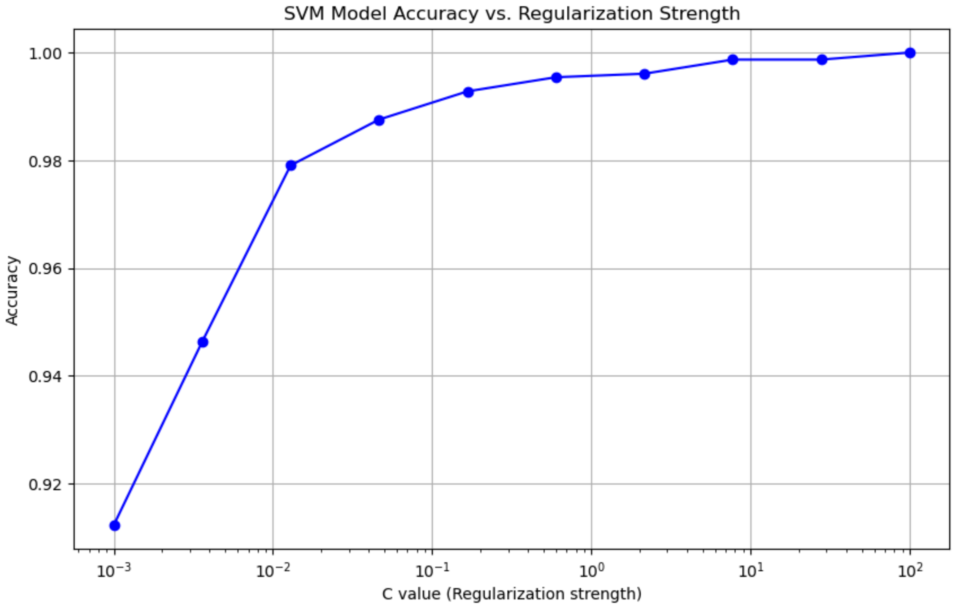
```
              precision    recall  f1-score   support

     Class 0       0.94      0.92      0.93       965
     Class 1       0.92      0.94      0.93       945

    accuracy                           0.93      1910
   macro avg       0.93      0.93      0.93      1910
weighted avg       0.93      0.93      0.93      1910
```

The screenshot above depicts the report of the three quantitative measures: precision, recall, and f1-score. The accuracy of our data is reported as well, implying that our model is 93% correct. We see that the values for precision and recall are all above 0.90 which means that our model is accurately predicting when to bet over/under. The f1-score appears to stay at a constant 0.93 despite what class it is a part of. We decided to implement accuracy and F1-score as our measure of performance because it provides a quick and intuitive way to test our imbalance classes. Through accuracy we can test how correct our model is through all classifications, and through F1-score we can analyze the performance metrics for the positive class.

Analysis of Random Forest Model:

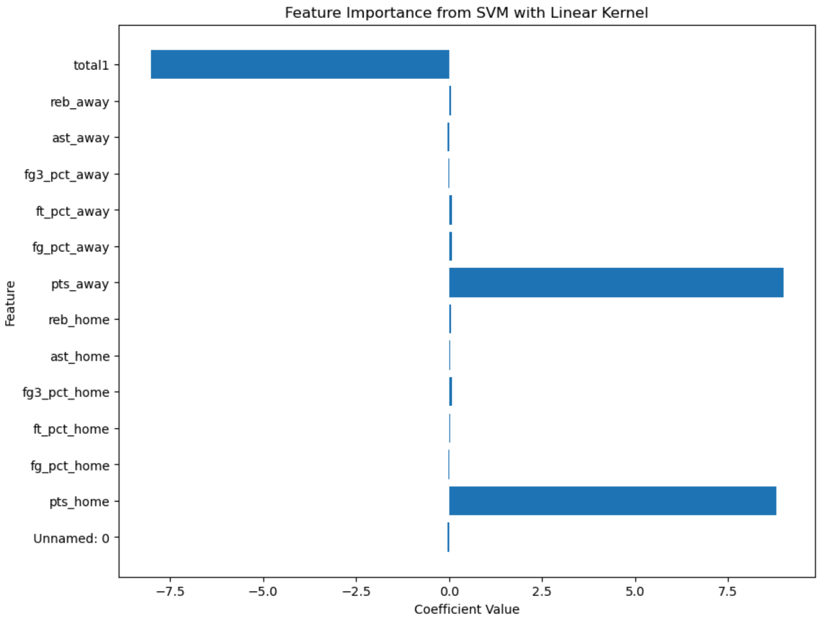
Although calculated earlier in the confusion matrix, after running our Random Forest model, we attain the following values within the classification report. We wanted to include accuracy and F1-score to keep a consistent indicator throughout all our models on how well they performed. Immediately, from a first glance, we are able to conclude that the overall accuracy and f1-score was 0.93, which was significantly higher than the accuracy we attained in the logistic regression model. This indicates that our random forest model is more suitable and appropriate in the context of our project. We conclude that the random forest model produces more accurate results primarily due to factors such as higher dimensionality and multicollinearity. If we parse through our dataset, it appears that there are a large amount of input variables that are highly related to one another. If we look at the heatmap presented in section three, we have roughly 18 variables that all play an impact on how the model makes decisions. We believe that the regression model conducted earlier was highly influenced by the noise in the dataset, but the random forest model does a better job in terms of reducing the noise and is robust to predictor variables being highly related to one another. Additionally, Class 0 represents accurately predicting when to bet under the over-under and Class 1 represents accurately predicting when to bet over the over-under, both of which have relatively high accuracy. Another potential factor that led to the high accuracy of our random forest model is due to the algorithm's nature of estimating what variables are important in the classification. This may be due to node impurity within trees as the quality of a variable is split using MSE which determines the amount of variance within each node. The greater the reduction in variance correlates to having greater importance.

Visualization of Support Vector Machine:



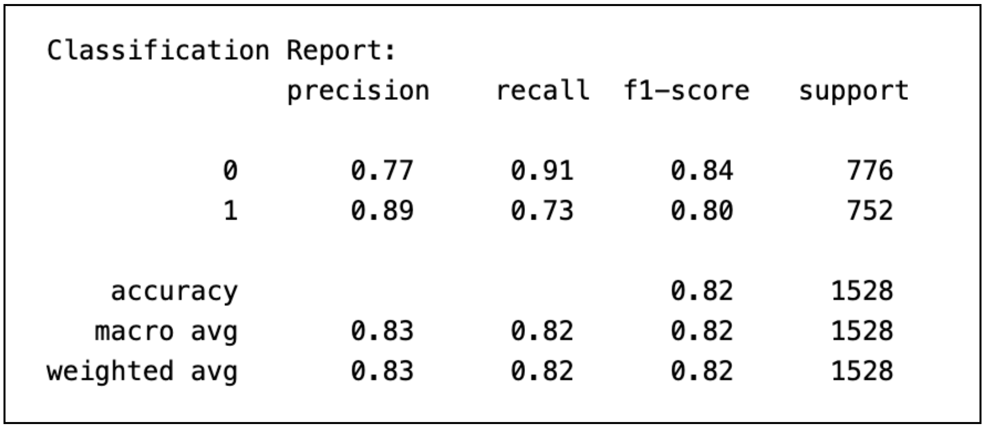
The line graph provides a visual representation of the accuracy of our Support Vector Machine model with relation to different values of *C* that serves as the regularization strength. This visualization completes a Support Vector Machine model for each C and evaluates the accuracy against the test set. C represents the error penalty when calculating the sum of the squared errors in the Support Vector

Machine’s optimization formulation. The general upward trend in terms of the curve signifies that as the regularization strength increases, our model attains a higher accuracy. This is helpful as we can now conclude we need to make our regularization strength relatively high in order to attain maximum accuracy.



The second visualization for the Support Vector Machine deals with a linear kernel. A linear kernel can be defined as the simplest form of kernels available for Support Vector Machine that simply computes the dot product of two data vectors. In this case, the linear kernel would be most applicable primarily because we have a large number of features, and the importance for each feature are reweighted accordingly. Based on the importance of the feature, the coefficient value for the respective feature is adjusted. In the image presented above, we can see that the Support Vector Machine prioritizes points scored at home, points scored away, and the total number of points a given NBA team scores. These 3 features have a significantly larger coefficient value than the other one’s which indicate that their importance is larger as well.

Quantitative Metrics of Support Vector Machine:



```
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.91      0.84       776
           1       0.89      0.73      0.80       752

    accuracy                           0.82      1528
   macro avg       0.83      0.82      0.82      1528
weighted avg       0.83      0.82      0.82      1528
```

The screenshot above depicts the report of the three quantitative measures: precision, recall, and f1-score. We can see that the accuracy for the Support Vector Machine model lies at 0.82 which indicates that our model is mostly accurate. Additionally, the precision and recall values for Class 0 and Class 1 roughly lie within the same range which implies that our model is predicting when to bet over the over-under or under the over-under relatively equally. Although, through running multiple trials to train the model, we can attempt to approve accuracy. We used accuracy and F1-score as quantitative measures because we wanted to keep it constant across all three models so we can compare the values.

Analysis of Support Vector Machine:

We believe that the Support Vector Machine was the most suitable algorithm for our project due to its nature of being compatible with binary classification tasks. This model performs well with datasets that have a large amount of dimensions. The first visualization, which is a line graph of the SVM model accuracy against different values of regularization strength C, shows the model's sensitivity to the regularization parameter. This graph shows a completely upward trend, which shows that as the regularization strength increases, so does the model's accuracy. This means that the model will benefit from a higher penalty on the error training, which will prevent overfitting. Further on, the evening out of the curve shows that past a certain point, increasing C does not significantly increase accuracy, so there was an optimal range for C. The second visualization, which is an SVM with a linear kernel, is justified by the large number of features in the dataset, so there is a more nuanced adjustment of feature weights. The displayed bar chart shows that the SVM has larger coefficient values to features related to points scored at home, away, and also the total number of points scored by the team. These features are shown to have a larger impact on the predictions, which is probably a result of the high scores in the NBA and betting. Lastly, the classification report shows multiple things: the precision, recall, and F-1 score for the two classes. The overall accuracy was 0.82, which suggests that the SVM model is correct in its predictions. Both of these classes have similar precision and recall values, which shows a balanced performance by the model in predicting the over and under results.

Comparison of Algorithms/Models:

Machine Learning Model/Algorithm	**Accuracy, F1-Score ((Class 0 + Class 1) / 2)**
Logistic Regression	0.58, 0.37
Random Forest	0.93, 0.93
Support Vector Machine	0.82, 0.82

After completing all three algorithms, it is important to note that each algorithm was different in terms of its approach and the dataset that was used. For instance, in the first model, we tried using logistic regression to predict the effect of an individual team's performance on the outcome of whether the next game hits over-under or not. This used only one team's performance as a factor in whether they would hit the over or not. Each team was made a dataframe which tracked their performance out of the given dataset for the given time period. We found the accuracy to be 0.58 and the precision and recall to be above 0.7. This leads us to believe that while there is some influence by an individual team's performance, it is not alone enough, at least in logistic regression, to be able to confidently predict an outcome. This may be due to the logistic regressions nature of operating with linear relationships and features that are not correlated.

We can observe in the dataset that all of our data does not follow a linear relationship model, and that features are related to one another.

For the random forest model, we wanted to implement a model that is capable of handling complex non-linear relationships between the features and the target. Another added benefit of implementing the random forest model is that it can handle feature importance intrusively. In other words, the model is able to provide insights regarding which features are the most important when predicting the target variable. This is done through node impurity within trees. Based on the classification report, we can see that this model produces the highest accuracy at 0.93, with an F1-score of 0.93 as well. This algorithm appears to be the most consistent in terms of performance evaluation, and we conclude that this may be due to the model's nature of eliminating noise and handling complex variables that might exhibit collinearity. As we are dealing with "over" vs. "under", which is a binary value, random forest appeared to be suitable for our project.

Lastly, we can also compare our results to the later Support Vector Model we attempted as well. This model maximizes the radial distance between the data to draw conclusions on classification. More preprocessing was needed, and this was done by computing the rolling averages of the box score statistics as well as their change over the same time period. There were a total of 25 features and were fed into a Support Vector Machine classifier to train and predict the outcome of the next game. Based on the classification report, we can see that we attained an accuracy of 0.82 with an F1-score that ranges above 0.80 as well. Although we attained a decent accuracy, one of our theories for error is the excess usage of similar and covariant features. For example, the overlap between some of the categories in the box score could have contributed to incorrect calculations from the model. One of the benefits of the Support Vector model is that we capture complex relations through the kernel. However, it appears that the random forest model on the whole attained the highest accuracy and F1-score making it the most suitable for our project.

Next Steps:

While these were fundamentally good starts at trying to solve this problem, obviously much more work and supervision is needed. Particularly, the next steps are to tune the hyperparameters of these models and examine them further for accuracy and evaluation. One specific step would be for the SVM model, to conduct a grid search to find the optimal kernel and gamma inputs. Additionally, more variance needs to be studied between all of the models. It is well known NBA statistics are highly variable and stochastic at times. Feature importance and feature selection must be paramount in selecting the most correct and effective algorithm. Moreover, since we are dealing with monetary funds as our project revolves around sports betting, we would want to maximize the accuracy of our model so that user's don't lose their money. This can be done through many methods such as identifying apparent collinearity within the features of our variable. Additionally, when comparing our model with other studies regarding NBA sports betting, we found that most models referred to the PCA method. This approach resulted in high accuracy over multiple trails. One of the first steps we can do this is through standardizing each data point so that the mean is equal to 0 and the standard deviation is equal to 1. Some of the other potential steps we can follow if we had additional time to refine our current progress is by applying regularization techniques such as ridge regression to prevent any overfitting patterns within the data. Although, it is important to note that due to unforeseen circumstances, there are days in sports in which error is inevitable. Whether it be due to player performance or other extraneous factors, having some room for error is guaranteed, however, we must prioritize minimizing such errors.

Section 5: References (IEEE Format)

[1] C. Osken and C. Onay, “Predicting the winning team in basketball: A novel approach,” Heliyon, vol. 8, no. 12, p. e12189, Dec. 2022, doi: https://doi.org/10.1016/j.heliyon.2022.e12189.

[2] J. Kuehn, “Accounting for Complementary Skill Sets: Evaluating Individual Marginal Value to a Team in the National Basketball Association,” Economic Inquiry, vol. 55, no. 3, pp. 1556–1578, Apr. 2017, doi: https://doi.org/10.1111/ecin.12451.

[3] M. Beckler, H. Wang, and M. Papamichael, “NBA Oracle,” 2013. Available: https://www.mbeckler.org/coursework/2008-2009/10701_report.pdf