Boston Celtics Championship Run (2023-2024 Season)

Asia Osimeh, Orlando Marin, Tatiana Eng

Topics of Discussion

- Introductions
- Background Information and Project Goals
- Dataset Selection and Data Preparation
- Data Analysis, Visualizations, and Code
- Analysis Limitations
- Main Takeaways
- Questions

Background Information and Project Goals

How does an NBA Season Work?

- NBA consists of 30 teams split into 2 conferences, East and West, each with 15 teams
- Each team plays 82 games during the regular season - 41 at home and 41 away
- Each game is 48 minutes, split into four 12-minute quarters
- Three ways to score - Free throws (1 point), 2 pt shot, 3 pt shot
- The team that scores the most points at the end of each game wins
- At the end of the regular season, the top 8 teams from each conference (based on Win - Loss record) make the playoffs and compete for the championship
- Celtics record last season (2023 - 2024): 64 wins, 18 losses (0.780 win percentage)

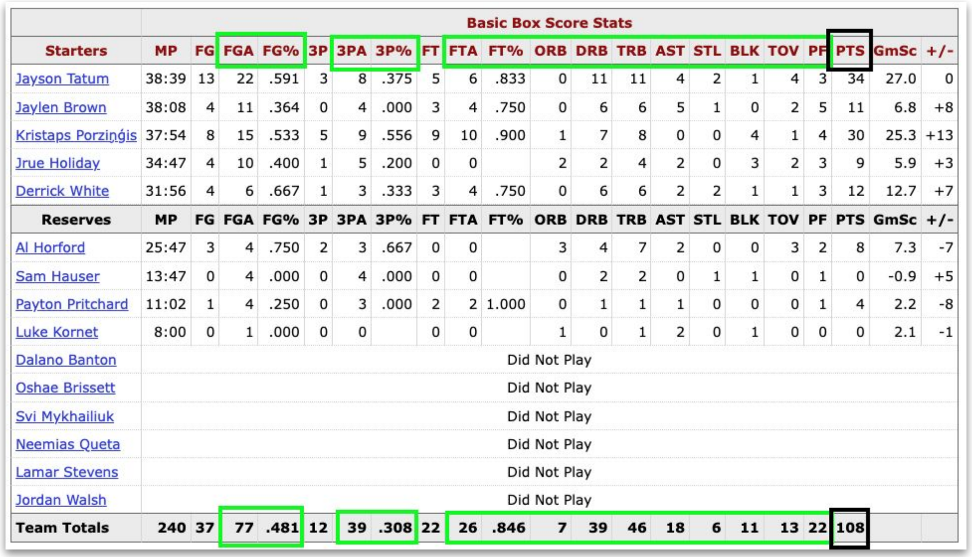
Questions our Project Answers

1. How accurately can we predict the number of points scored by the Celtics and their opponents using basic box score team stats?
2. How accurately can we predict how many games the Celtics won in the 2023-2024 regular season?
3. Which statistics have the greatest impact on winning NBA games?

Dataset Selection and Data Preparation

Dataset Selection

- Dataset is from basketballreference.com
- Filtered specifically to the Boston Celtics game data from the 2023 - 2024 season
- Data was not in a format we could use, so we transferred the data to a spreadsheet and then downloaded as a CSV file



	Basic Box Score Stats																				
Starters	MP	FG	FGA	FG%	3P	3PA	3P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	GmSc	+/-
Jayson Tatum	38:39	13	22	.591	3	8	.375	5	6	.833	0	11	11	4	2	1	4	3	34	27.0	0
Jaylen Brown	38:08	4	11	.364	0	4	.000	3	4	.750	0	6	6	5	1	0	2	5	11	6.8	+8
Kristaps Porziņģis	37:54	8	15	.533	5	9	.556	9	10	.900	1	7	8	0	0	4	1	4	30	25.3	+13
Jrue Holiday	34:47	4	10	.400	1	5	.200	0	0		2	2	4	2	0	3	2	3	9	5.9	+3
Derrick White	31:56	4	6	.667	1	3	.333	3	4	.750	0	6	6	2	2	1	1	3	12	12.7	+7
Reserves	MP	FG	FGA	FG%	3P	3PA	3P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	GmSc	+/-
Al Horford	25:47	3	4	.750	2	3	.667	0	0		3	4	7	2	0	0	3	2	8	7.3	-7
Sam Hauser	13:47	0	4	.000	0	4	.000	0	0		0	2	2	0	1	1	0	1	0	-0.9	+5
Payton Pritchard	11:02	1	4	.250	0	3	.000	2	2	1.000	0	1	1	1	0	0	0	1	4	2.2	-8
Luke Kornet	8:00	0	1	.000	0	0		0	0		1	0	1	2	0	1	0	0	0	2.1	-1
Dalano Banton	Did Not Play																				
Oshae Brissett	Did Not Play																				
Svi Mykhailiuk	Did Not Play																				
Neemias Queta	Did Not Play																				
Lamar Stevens	Did Not Play																				
Jordan Walsh	Did Not Play																				
Team Totals	**240**	**37**	**77**	**.481**	**12**	**39**	**.308**	**22**	**26**	**.846**	**7**	**39**	**46**	**18**	**6**	**11**	**13**	**22**	**108**		

Dataset (online)
Dataset (spreadsheet)

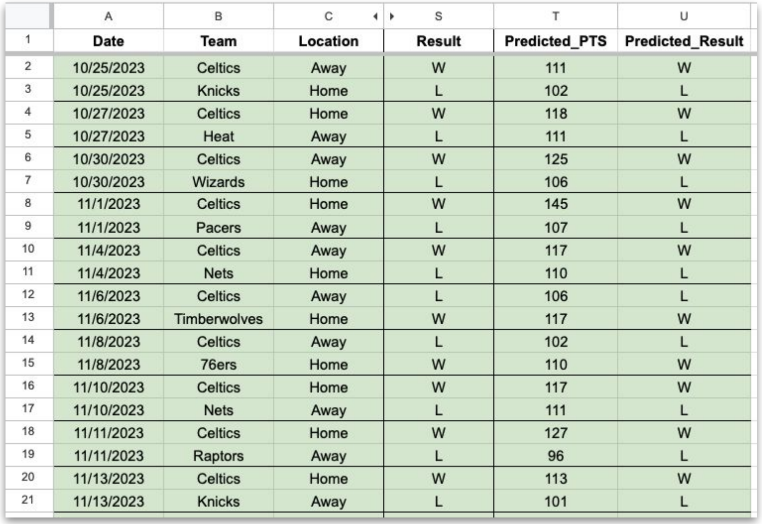
Basketball Statistics used to Predict Points

- FGA or FG% - Field Goal Attempts or Field Goal Percentage
- 3PA or 3P% - 3 Point Attempts or 3 Point Percentage
- FTA or FT% - Free Throw Attempts or Free Throw Percentage
- ORB - Offensive Rebounds
- DRB - Defensive Rebounds
- TRB - Total Rebounds
- AST - Assists
- STL - Steals
- BLK - Blocks
- TOV - Turnovers
- PF - Personal Fouls
- PTS - Points (what we're predicting)

NBA Statistics Definitions

Columns we Added to our Dataset

- Location - Home or Away
- Result - Win (W) or Loss (L)
- Predicted_PTS - rounded to the nearest whole number
- Predicted_Result - Win (W) or Loss (L)

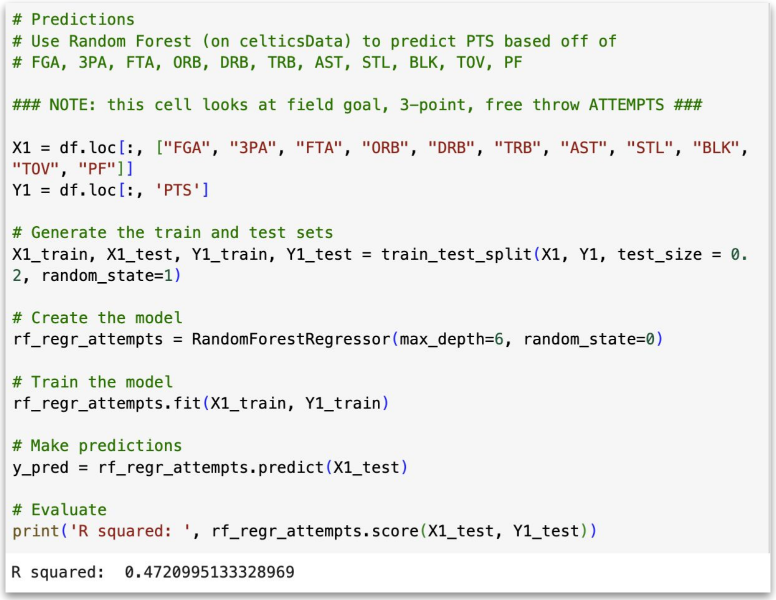


	A	B	C	S	T	U
1	**Date**	**Team**	**Location**	**Result**	**Predicted_PTS**	**Predicted_Result**
2	10/25/2023	Celtics	Away	W	111	W
3	10/25/2023	Knicks	Home	L	102	L
4	10/27/2023	Celtics	Home	W	118	W
5	10/27/2023	Heat	Away	L	111	L
6	10/30/2023	Celtics	Away	W	125	W
7	10/30/2023	Wizards	Home	L	106	L
8	11/1/2023	Celtics	Home	W	145	W
9	11/1/2023	Pacers	Away	L	107	L
10	11/4/2023	Celtics	Away	W	117	W
11	11/4/2023	Nets	Home	L	110	L
12	11/6/2023	Celtics	Away	L	106	L
13	11/6/2023	Timberwolves	Home	W	117	W
14	11/8/2023	Celtics	Away	L	102	L
15	11/8/2023	76ers	Home	W	110	W
16	11/10/2023	Celtics	Home	W	117	W
17	11/10/2023	Nets	Away	L	111	L
18	11/11/2023	Celtics	Home	W	127	W
19	11/11/2023	Raptors	Away	L	96	L
20	11/13/2023	Celtics	Home	W	113	W
21	11/13/2023	Knicks	Away	L	101	L

Dataset

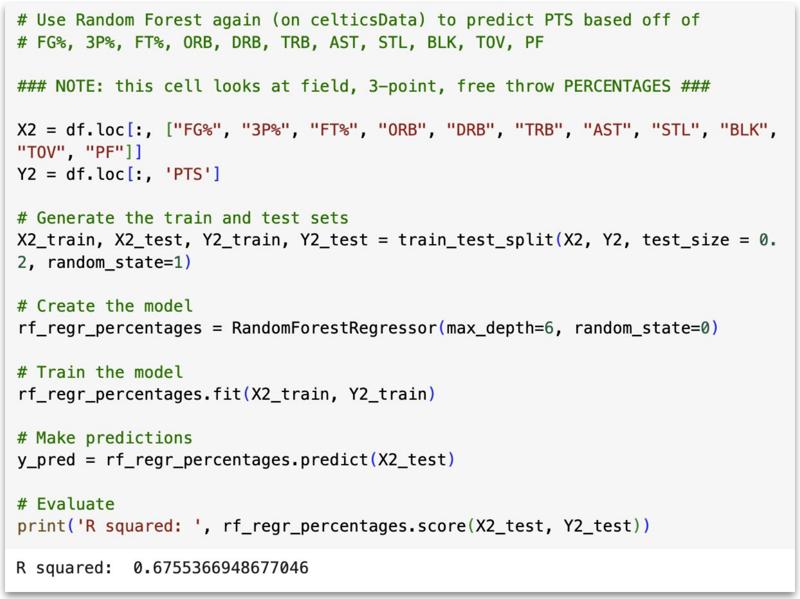
Data Analysis, Visualizations, and Code

Prediction Models Using Random Forest



```
# Predictions
# Use Random Forest (on celticsData) to predict PTS based off of
# FGA, 3PA, FTA, ORB, DRB, TRB, AST, STL, BLK, TOV, PF

### NOTE: this cell looks at field goal, 3-point, free throw ATTEMPTS ###

X1 = df.loc[:, ["FGA", "3PA", "FTA", "ORB", "DRB", "TRB", "AST", "STL", "BLK",
"TOV", "PF"]]
Y1 = df.loc[:, 'PTS']

# Generate the train and test sets
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X1, Y1, test_size = 0.
2, random_state=1)

# Create the model
rf_regr_attempts = RandomForestRegressor(max_depth=6, random_state=0)

# Train the model
rf_regr_attempts.fit(X1_train, Y1_train)

# Make predictions
y_pred = rf_regr_attempts.predict(X1_test)

# Evaluate
print('R squared: ', rf_regr_attempts.score(X1_test, Y1_test))
```

```
R squared:  0.4720995133328969
```



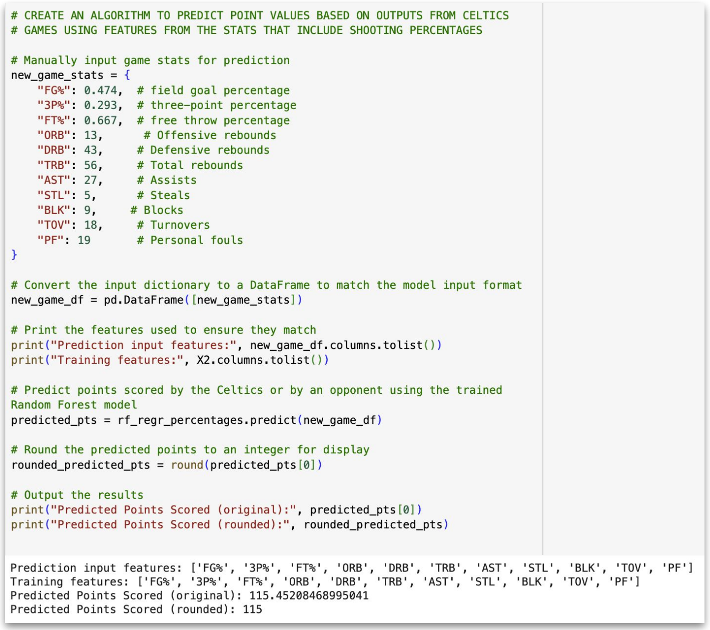
```
# Use Random Forest again (on celticsData) to predict PTS based off of
# FG%, 3P%, FT%, ORB, DRB, TRB, AST, STL, BLK, TOV, PF

### NOTE: this cell looks at field, 3-point, free throw PERCENTAGES ###

X2 = df.loc[:, ["FG%", "3P%", "FT%", "ORB", "DRB", "TRB", "AST", "STL", "BLK",
"TOV", "PF"]]
Y2 = df.loc[:, 'PTS']

# Generate the train and test sets
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X2, Y2, test_size = 0.
2, random_state=1)

# Create the model
rf_regr_percentages = RandomForestRegressor(max_depth=6, random_state=0)

# Train the model
rf_regr_percentages.fit(X2_train, Y2_train)

# Make predictions
y_pred = rf_regr_percentages.predict(X2_test)

# Evaluate
print('R squared: ', rf_regr_percentages.score(X2_test, Y2_test))
```

```
R squared:  0.6755366948677046
```

Google Colab

Predicting Points per Game

- We used the Random Forest Model with Shooting Percentages to predict the point scored in each game.
- Points were rounded up or down to the nearest whole number
- In the event of a predicted tie (1 instance), we used the non-rounded point total to determine who won and lost the game

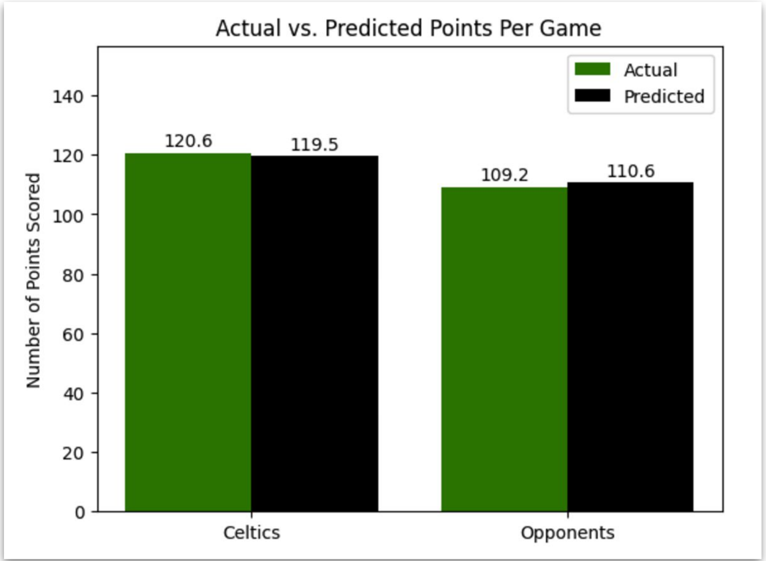


```
# CREATE AN ALGORITHM TO PREDICT POINT VALUES BASED ON OUTPUTS FROM CELTICS
# GAMES USING FEATURES FROM THE STATS THAT INCLUDE SHOOTING PERCENTAGES

# Manually input game stats for prediction
new_game_stats = {
    "FG%": 0.474,  # field goal percentage
    "3P%": 0.293,  # three-point percentage
    "FT%": 0.667,  # free throw percentage
    "ORB": 13,      # Offensive rebounds
    "DRB": 43,     # Defensive rebounds
    "TRB": 56,     # Total rebounds
    "AST": 27,     # Assists
    "STL": 5,      # Steals
    "BLK": 9,     # Blocks
    "TOV": 18,     # Turnovers
    "PF": 19       # Personal fouls
}

# Convert the input dictionary to a DataFrame to match the model input format
new_game_df = pd.DataFrame([new_game_stats])

# Print the features used to ensure they match
print("Prediction input features:", new_game_df.columns.tolist())
print("Training features:", X2.columns.tolist())

# Predict points scored by the Celtics or by an opponent using the trained
Random Forest model
predicted_pts = rf_regr_percentages.predict(new_game_df)

# Round the predicted points to an integer for display
rounded_predicted_pts = round(predicted_pts[0])

# Output the results
print("Predicted Points Scored (original):", predicted_pts[0])
print("Predicted Points Scored (rounded):", rounded_predicted_pts)
```

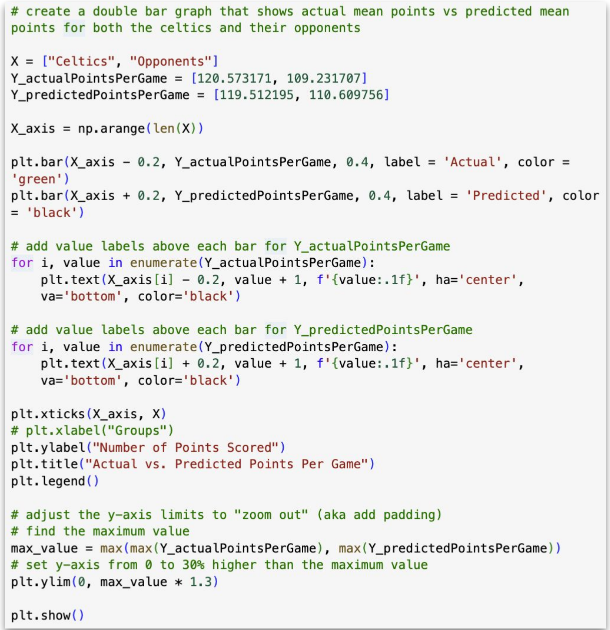
```
Prediction input features: ['FG%', '3P%', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF']
Training features: ['FG%', '3P%', 'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF']
Predicted Points Scored (original): 115.45208468995041
Predicted Points Scored (rounded): 115
```

Google Colab

How Accurate Were Our Points Predictions?



*Mean Absolute Error when predicting Celtics' and opponents' points per game: 3.37

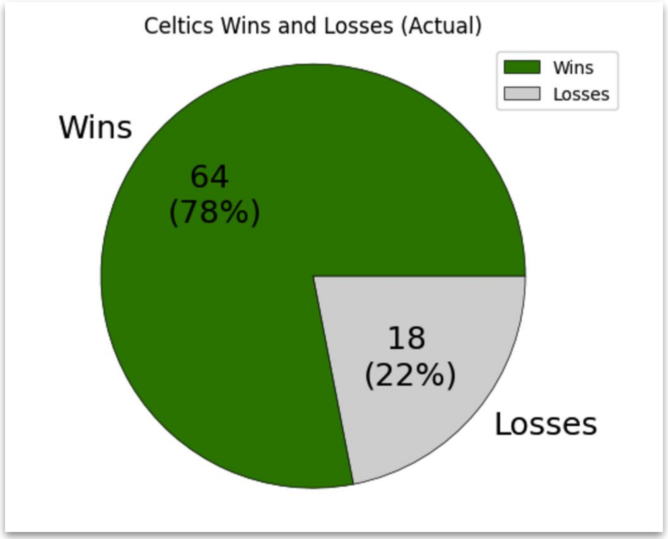


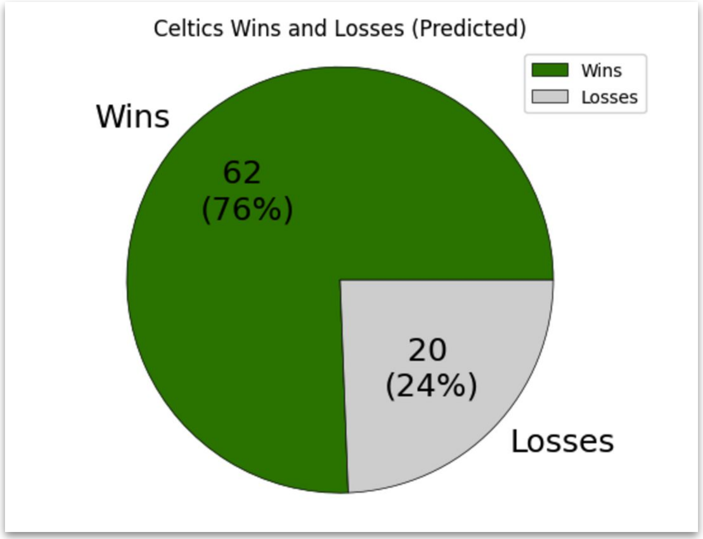
```
# create a double bar graph that shows actual mean points vs predicted mean
points for both the celtics and their opponents

X = ["Celtics", "Opponents"]
Y_actualPointsPerGame = [120.573171, 109.231707]
Y_predictedPointsPerGame = [119.512195, 110.609756]

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, Y_actualPointsPerGame, 0.4, label = 'Actual', color =
'green')
plt.bar(X_axis + 0.2, Y_predictedPointsPerGame, 0.4, label = 'Predicted', color
= 'black')

# add value labels above each bar for Y_actualPointsPerGame
for i, value in enumerate(Y_actualPointsPerGame):
    plt.text(X_axis[i] - 0.2, value + 1, f'{value:.1f}', ha='center',
    va='bottom', color='black')

# add value labels above each bar for Y_predictedPointsPerGame
for i, value in enumerate(Y_predictedPointsPerGame):
    plt.text(X_axis[i] + 0.2, value + 1, f'{value:.1f}', ha='center',
    va='bottom', color='black')

plt.xticks(X_axis, X)
# plt.xlabel("Groups")
plt.ylabel("Number of Points Scored")
plt.title("Actual vs. Predicted Points Per Game")
plt.legend()

# adjust the y-axis limits to "zoom out" (aka add padding)
# find the maximum value
max_value = max(max(Y_actualPointsPerGame), max(Y_predictedPointsPerGame))
# set y-axis from 0 to 30% higher than the maximum value
plt.ylim(0, max_value * 1.3)

plt.show()
```

Google Colab
Dataset

How Many Wins Did We Accurately Predict?

Accurately predicted the results of 95.12% (78/82) of Celtics’ games

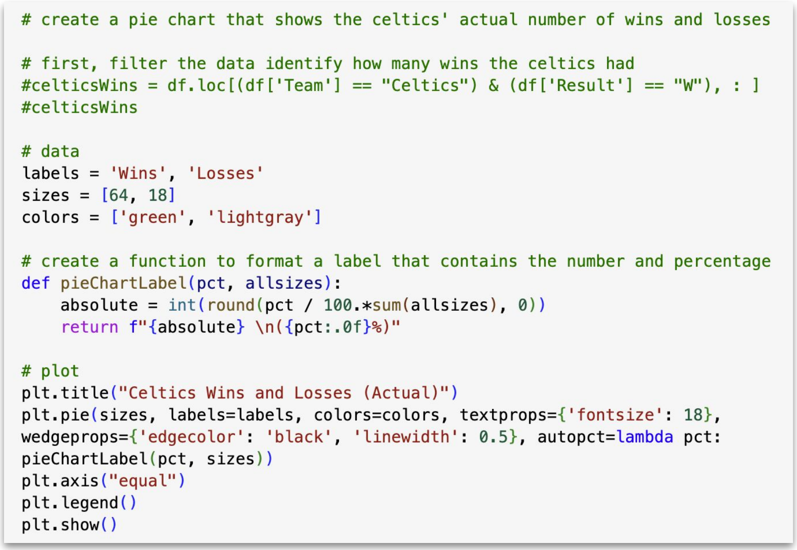




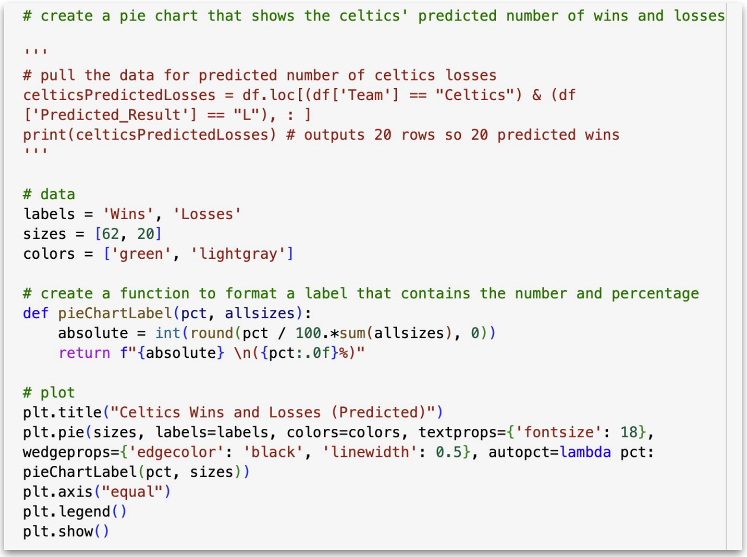
3 wins incorrectly predicted as a loss (-3 wins), 1 loss incorrectly predicted as a win (+1 win)
Predicted win differential of (-2)

Google Colab
Dataset

Creating the Pie Charts



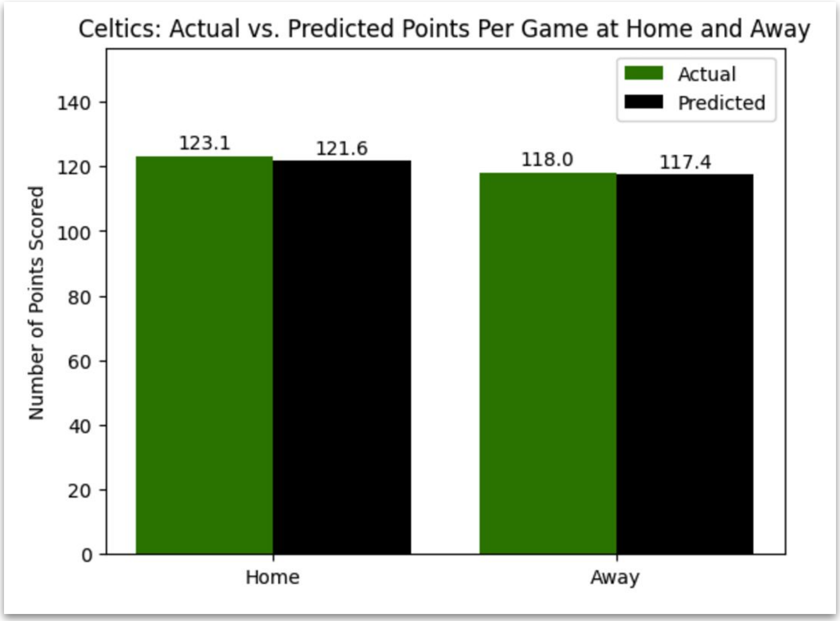
```python
# create a pie chart that shows the celtics' actual number of wins and losses

# first, filter the data identify how many wins the celtics had
#celticsWins = df.loc[(df['Team'] == "Celtics") & (df['Result'] == "W"), : ]
#celticsWins

# data
labels = 'Wins', 'Losses'
sizes = [64, 18]
colors = ['green', 'lightgray']

# create a function to format a label that contains the number and percentage
def pieChartLabel(pct, allsizes):
    absolute = int(round(pct / 100.*sum(allsizes), 0))
    return f"{absolute} \n({pct:.0f}%)"

# plot
plt.title("Celtics Wins and Losses (Actual)")
plt.pie(sizes, labels=labels, colors=colors, textprops={'fontsize': 18},
wedgeprops={'edgecolor': 'black', 'linewidth': 0.5}, autopct=lambda pct:
pieChartLabel(pct, sizes))
plt.axis("equal")
plt.legend()
plt.show()
```

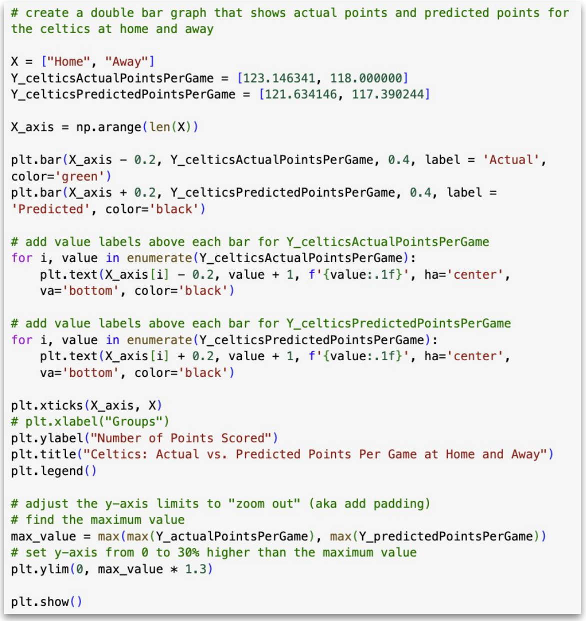


```python
# create a pie chart that shows the celtics' predicted number of wins and losses

'''
# pull the data for predicted number of celtics losses
celticsPredictedLosses = df.loc[(df['Team'] == "Celtics") & (df
['Predicted_Result'] == "L"), : ]
print(celticsPredictedLosses) # outputs 20 rows so 20 predicted wins
'''

# data
labels = 'Wins', 'Losses'
sizes = [62, 20]
colors = ['green', 'lightgray']

# create a function to format a label that contains the number and percentage
def pieChartLabel(pct, allsizes):
    absolute = int(round(pct / 100.*sum(allsizes), 0))
    return f"{absolute} \n({pct:.0f}%)"

# plot
plt.title("Celtics Wins and Losses (Predicted)")
plt.pie(sizes, labels=labels, colors=colors, textprops={'fontsize': 18},
wedgeprops={'edgecolor': 'black', 'linewidth': 0.5}, autopct=lambda pct:
pieChartLabel(pct, sizes))
plt.axis("equal")
plt.legend()
plt.show()
```

Google Colab

Celtics When Playing at Home vs Away

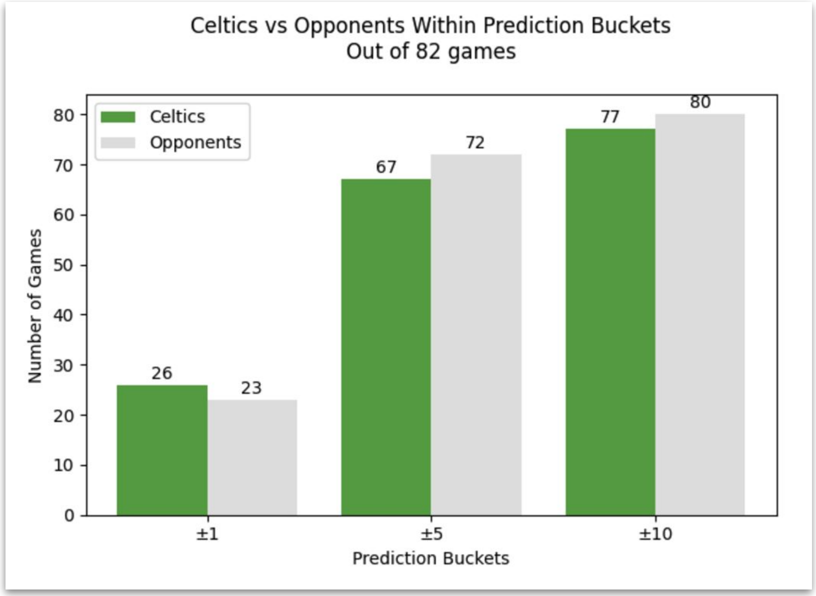


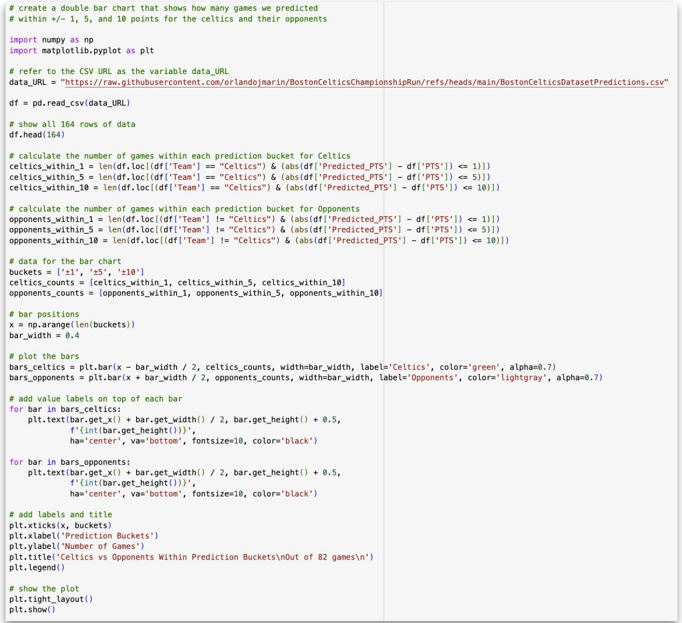


```
# create a double bar graph that shows actual points and predicted points for
the celtics at home and away

X = ["Home", "Away"]
Y_celticsActualPointsPerGame = [123.146341, 118.000000]
Y_celticsPredictedPointsPerGame = [121.634146, 117.390244]

X_axis = np.arange(len(X))

plt.bar(X_axis - 0.2, Y_celticsActualPointsPerGame, 0.4, label = 'Actual',
color='green')
plt.bar(X_axis + 0.2, Y_celticsPredictedPointsPerGame, 0.4, label =
'Predicted', color='black')

# add value labels above each bar for Y_celticsActualPointsPerGame
for i, value in enumerate(Y_celticsActualPointsPerGame):
    plt.text(X_axis[i] - 0.2, value + 1, f'{value:.1f}', ha='center',
    va='bottom', color='black')

# add value labels above each bar for Y_celticsPredictedPointsPerGame
for i, value in enumerate(Y_celticsPredictedPointsPerGame):
    plt.text(X_axis[i] + 0.2, value + 1, f'{value:.1f}', ha='center',
    va='bottom', color='black')

plt.xticks(X_axis, X)
# plt.xlabel("Groups")
plt.ylabel("Number of Points Scored")
plt.title("Celtics: Actual vs. Predicted Points Per Game at Home and Away")
plt.legend()

# adjust the y-axis limits to "zoom out" (aka add padding)
# find the maximum value
max_value = max(max(Y_actualPointsPerGame), max(Y_predictedPointsPerGame))
# set y-axis from 0 to 30% higher than the maximum value
plt.ylim(0, max_value * 1.3)

plt.show()
```

Google Colab

Categorizing Our Points Predictions

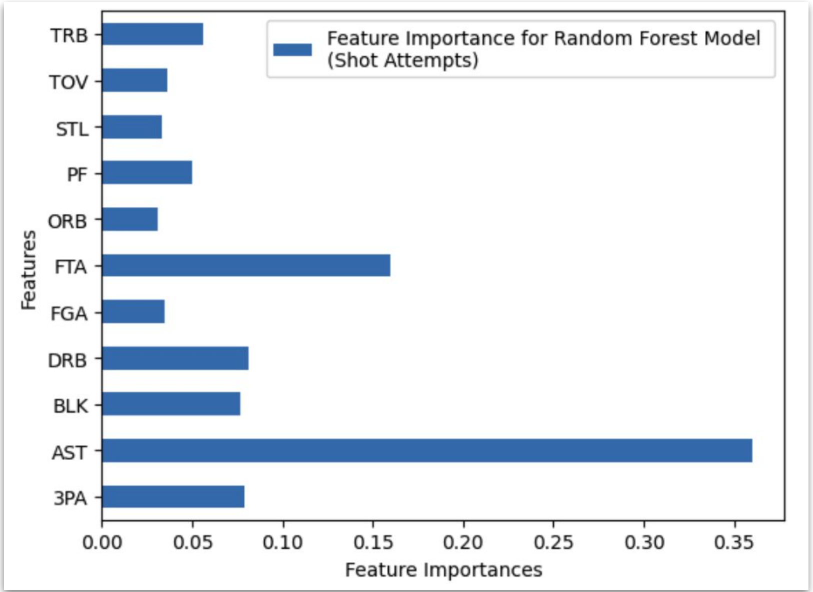




```python
# create a double bar chart that shows how many games we predicted
# within +/- 1, 5, and 10 points for the celtics and their opponents

import numpy as np
import matplotlib.pyplot as plt

# refer to the CSV URL as the variable data_URL
data_URL = "https://raw.githubusercontent.com/orlandojmarin/BostonCelticsChampionshipRun/refs/heads/main/BostonCelticsDatasetPredictions.csv"

df = pd.read_csv(data_URL)

# show all 164 rows of data
df.head(164)

# calculate the number of games within each prediction bucket for Celtics
celtics_within_1 = len(df.loc[(df['Team'] == "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 1)])
celtics_within_5 = len(df.loc[(df['Team'] == "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 5)])
celtics_within_10 = len(df.loc[(df['Team'] == "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 10)])

# calculate the number of games within each prediction bucket for Opponents
opponents_within_1 = len(df.loc[(df['Team'] != "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 1)])
opponents_within_5 = len(df.loc[(df['Team'] != "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 5)])
opponents_within_10 = len(df.loc[(df['Team'] != "Celtics") & (abs(df['Predicted_PTS'] - df['PTS']) <= 10)])

# data for the bar chart
buckets = ['±1', '±5', '±10']
celtics_counts = [celtics_within_1, celtics_within_5, celtics_within_10]
opponents_counts = [opponents_within_1, opponents_within_5, opponents_within_10]

# bar positions
x = np.arange(len(buckets))
bar_width = 0.4

# plot the bars
bars_celtics = plt.bar(x - bar_width / 2, celtics_counts, width=bar_width, label='Celtics', color='green', alpha=0.7)
bars_opponents = plt.bar(x + bar_width / 2, opponents_counts, width=bar_width, label='Opponents', color='lightgray', alpha=0.7)

# add value labels on top of each bar
for bar in bars_celtics:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.5,
             f'{int(bar.get_height())}',
             ha='center', va='bottom', fontsize=10, color='black')

for bar in bars_opponents:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.5,
             f'{int(bar.get_height())}',
             ha='center', va='bottom', fontsize=10, color='black')

# add labels and title
plt.xticks(x, buckets)
plt.xlabel('Prediction Buckets')
plt.ylabel('Number of Games')
plt.title('Celtics vs Opponents Within Prediction Buckets\nOut of 82 games\n')
plt.legend()

# show the plot
plt.tight_layout()
plt.show()
```

Google Colab

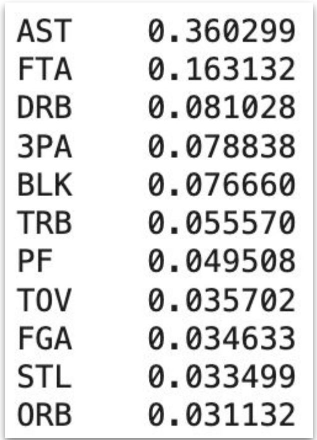
Feature Importance (1st Random Forest Model)



Google Colab

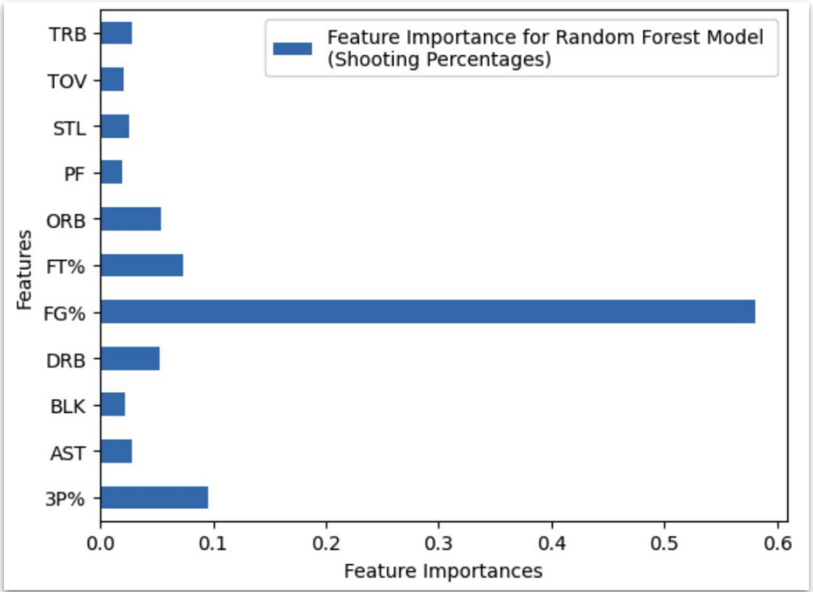
Feature Importances

R^2 when using shot attempts: 0.4721



```
AST     0.360299
FTA     0.163132
DRB     0.081028
3PA     0.078838
BLK     0.076660
TRB     0.055570
PF      0.049508
TOV     0.035702
FGA     0.034633
STL     0.033499
ORB     0.031132
```

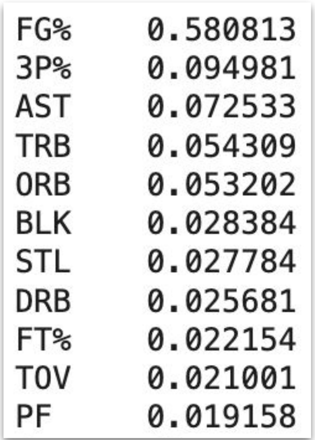
Feature Importance (2nd Random Forest Model)



Google Colab

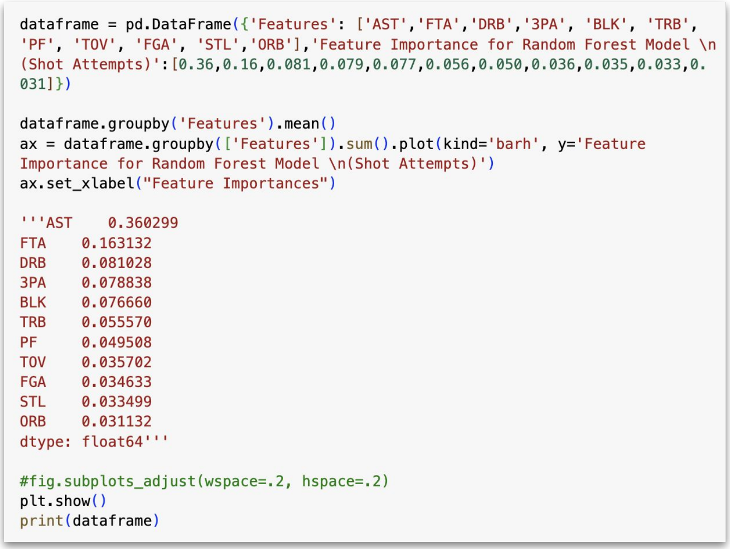
Feature Importances

R^2 when using shot attempts: 0.6755



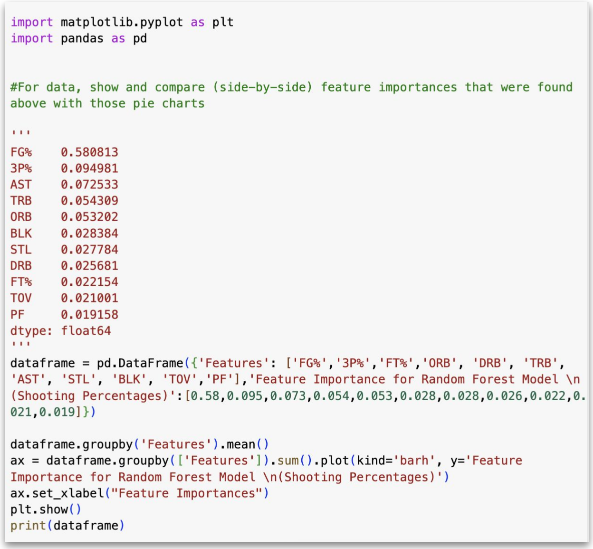
```
FG%     0.580813
3P%     0.094981
AST     0.072533
TRB     0.054309
ORB     0.053202
BLK     0.028384
STL     0.027784
DRB     0.025681
FT%     0.022154
TOV     0.021001
PF      0.019158
```

Creating Horizontal Bar Graphs to Show Feature Importance



```
dataframe = pd.DataFrame({'Features': ['AST','FTA','DRB','3PA', 'BLK', 'TRB',
'PF', 'TOV', 'FGA', 'STL','ORB'],'Feature Importance for Random Forest Model \n
(Shot Attempts)':[0.36,0.16,0.081,0.079,0.077,0.056,0.050,0.036,0.035,0.033,0.
031]})

dataframe.groupby('Features').mean()
ax = dataframe.groupby(['Features']).sum().plot(kind='barh', y='Feature
Importance for Random Forest Model \n(Shot Attempts)')
ax.set_xlabel("Feature Importances")

'''AST    0.360299
FTA    0.163132
DRB    0.081028
3PA    0.078838
BLK    0.076660
TRB    0.055570
PF     0.049508
TOV    0.035702
FGA    0.034633
STL    0.033499
ORB    0.031132
dtype: float64'''

#fig.subplots_adjust(wspace=.2, hspace=.2)
plt.show()
print(dataframe)
```

Shot attempts



```
import matplotlib.pyplot as plt
import pandas as pd


#For data, show and compare (side-by-side) feature importances that were found
above with those pie charts

'''
FG%    0.580813
3P%    0.094981
AST    0.072533
TRB    0.054309
ORB    0.053202
BLK    0.028384
STL    0.027784
DRB    0.025681
FT%    0.022154
TOV    0.021001
PF     0.019158
dtype: float64
'''
dataframe = pd.DataFrame({'Features': ['FG%','3P%','FT%','ORB', 'DRB', 'TRB',
'AST', 'STL', 'BLK', 'TOV','PF'],'Feature Importance for Random Forest Model \n
(Shooting Percentages)':[0.58,0.095,0.073,0.054,0.053,0.028,0.028,0.026,0.022,0.
021,0.019]})

dataframe.groupby('Features').mean()
ax = dataframe.groupby(['Features']).sum().plot(kind='barh', y='Feature
Importance for Random Forest Model \n(Shooting Percentages)')
ax.set_xlabel("Feature Importances")
plt.show()
print(dataframe)
```

Shooting percentage

Feature Importance & the Four Factors

- Shot Attempts Random Forest Model
  - Top 4 Features: Assists, Free Throw Attempts, Defensive Rebounds, and Three-Point Attempts
  - Impact: Highlights the importance of playmaking, free throw opportunities, rebounding, and three point shooting
- Shooting Percentages Random Forest Model
  - Top 4 Features: Field Goal Percentage, Three-Point Percentage, Assists, and Total Rebounds
  - Impact: Highlights the importance of shooting efficiency, playmaking, and rebounding
- Our Top 4 Factors to Win Games:
  - Shooting Percentage (Field Goal % and Three-Point %)
  - Assists
  - Free Throw Attempts
  - Rebounds (Defensive Rebounds and Total Rebounds)
- Dean Oliver's "Four Factors of Basketball Success" to Win Games:
  - Effective Field Goal Percentage
  - Turnover Percentage
  - Offensive and Defensive Rebound Percentage
  - Free Throws per Field Goal Attempt

Dean Oliver's Four Factors of Basketball Success

Analysis Limitations

Analysis Limitations

- Imbalance in Data Representation
  - The random forest model that we used was based solely on Celtics games during the 2023 - 2024 season. As a result, half of the data (82 out of 164 rows) is from the Celtics.
  - There is a risk of us more accurately predicting Celtics results as opposed to their opponents
  - Ideally, we would use a different model for all 30 teams, based on all of their games played, to further improve model performance
- Limited Feature Scope
  - The model only considers basic box score stats and doesn’t use advanced metrics like Effective Field Goal Percentage, Turnovers Committed per Possession, Offensive Rebounding Percentage, or Free Throw Rate
  - The model only considers in-game statistics, without accounting for external factors like player injuries, lineup changes, rest and practice days in between games, and travel, which can impact the outcome of each game
  - Including advanced metrics could improve accuracy
- Assumption of Independent Games
  - The model treats each game independently, ignoring things such as momentum across multiple games, which can be difficult to quantify

Main Takeaways

Main Takeaways

1. How accurately can we predict the number of points scored by the Celtics and their opponents using basic box score team stats?
    - We predicted 31.7% (26/82) of Celtics point totals and 28.0% (23/82) opponents point totals within 1 point of the actual result
    - We predicted 81.7% (67/82) of Celtics point totals and 87.8% (72/82) opponents point totals within 5 points of the actual result
2. How accurately can we predict how many games the Celtics won in the 2023-2024 regular season?
    - We accurately predicted the results of 95.1% of Celtics games (78/82)
    - Predicted win differential of (-2); 64 actual wins compared to 62 predicted wins
3. Which statistics have the greatest impact on winning NBA games?
    - Shooting Percentage (Field Goal % and Three-Point %)
    - Assists
    - Free Throw Attempts
    - Rebounds (Defensive Rebounds and Total Rebounds)

Questions?