Swish Insights

By: Brandon, Elvin, & Michael

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Project Selection Process

- What should we do?
- Sports?
- Basketball seems easy
- Regret
- Perseverance

Why should you care?

- Profitable
- Financial Application
- Entertainment
- Addicting

Completion Process

- Data collection & Scraping
- Preprocessing
- Custom Scoring
- CV Fold
- Machine Learning
- Outputting Model Prediction

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Basic Steps

1. Finding Necessary Data

Basic Steps

1. Finding Necessary Data
    a. Box score data from basketball-reference
    b. Odds for ML, Spreads, and Totals from Action-Network

Basic Steps

1. Finding Necessary Data
2. Saving Necessary Data
    a. Save box score datas as csvs
    b. Save odds data as excel files

```python
def save_box_scores(formatted_date_list, simple_games_df, url_base):
    # Create a main directory to hold all data before zipping
    main_folder = 'NBA_Box_Scores'
    os.makedirs(main_folder, exist_ok=True)
    for date in formatted_date_list:
        games_df = simple_games_df[simple_games_df['Date'] == date]
        for index, row in games_df.iterrows():
            home_abbr = row['Home']
            away_abbr = row['Away']
            game_folder = f"{date}/{away_abbr}@{home_abbr}"  # Folder name format: YYYYMMDD/Away@Home
            full_folder_path = os.path.join(main_folder, game_folder)
            os.makedirs(full_folder_path, exist_ok=True)

            # Format the URL
            formatted_url = f"{url_base}{date}0{home_abbr}.html"
            # Fetch and save box scores
            try:
                response = fetch_with_retry_after(formatted_url)
            # Save each team's box score in the specific game folder
                away_df,home_df = get_boxscore(response.text)
                away_df.to_csv(f"{full_folder_path}/away_team.csv", index=False)
                home_df.to_csv(f"{full_folder_path}/home_team.csv", index=False)
            except Exception as e:
                print(f"Error fetching data for URL {formatted_url}: {str(e)}")

    # Zip the entire directory
    with ZipFile(f"{main_folder}.zip", 'w') as zipf:
        for root, dirs, files in os.walk(main_folder):
            for file in files:
                zipf.write(os.path.join(root, file), os.path.relpath(os.path.join(root, file), os.path.join(main_folder, '..')))

url_base = "https://www.basketball-reference.com/boxscores/"

save_box_scores(formatted_date_list, simple_games_df, url_base)
```

Basic Steps

1. Finding Necessary Data
2. Saving Necessary Data
3. Transforming box scores into a statsheet
    a. Loop through dates and add box score data to a running statsheet of teams and players
    b. Functions to 1) update from box score, 2) add to existing value, 3) create a new value
    c. Creates a df for each day, indexed by team/ player name, then put into folders by date

Main Loop

```
## creates cumul team stats for each date
all_team_stats = pd.DataFrame()
main_folder = "NBA_Team_Statsheet"
os.makedirs(main_folder, exist_ok = True)
for date in formatted_date_list:
    date_folder = f"{date}"
    full_folder_path = os.path.join(main_folder, date_folder)
    os.makedirs(full_folder_path, exist_ok = True)
    games_df = simple_games_df[simple_games_df['Date'] == date]
    for index, row in games_df.iterrows():
        # Format the URL with the current game's home team abbreviation
        home_abbr = row['Home']
        away_abbr = row['Away']

        all_team_stats = update_team_from_boxscore(all_team_stats,date,away_abbr,home_abbr)
    all_team_stats.to_csv(f"{full_folder_path}/statsheet.csv")
```

Supporting Function

```python
def update_team_from_boxscore(data,date, away, home ): # path is  NBA_Box_Scores/date/away@home
    #data = pd.DataFrame()
    base_directory = "NBA_Box_Scores"
    directory_path = f"{base_directory}/{date}/{away}@{home}"
    boxscore_files = [f for f in os.listdir(directory_path) if f.endswith('.csv')]
    for file_name in boxscore_files:
        if file_name == "away_team.csv":
            team_name = away
        elif file_name == "home_team.csv":
            team_name = home
        else:
            print("error no csv found")
            continue
        file_path = os.path.join(directory_path, file_name)
        df = pd.read_csv(file_path)
        team_stats_row = df.iloc[-1]
        if team_name in data.index:
            data = update_running_team_averages(data, team_name, team_stats_row)
        else:
            # Add new team
            data = add_new_team(data,team_name, team_stats_row)

    return data
```

Basic Steps

1. Finding Necessary Data
2. Saving Necessary Data
3. Transforming box scores into a statsheet
4. Organizing Data
    a. Restructuring data so each row is each game played for target team
    b. Including opponent data in this row as well, as well as betting data
    c. Creating new variables tracking L5 or L10 averages

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

What Data Are we Looking At?

Basketball Season

What Data Are we Looking At?

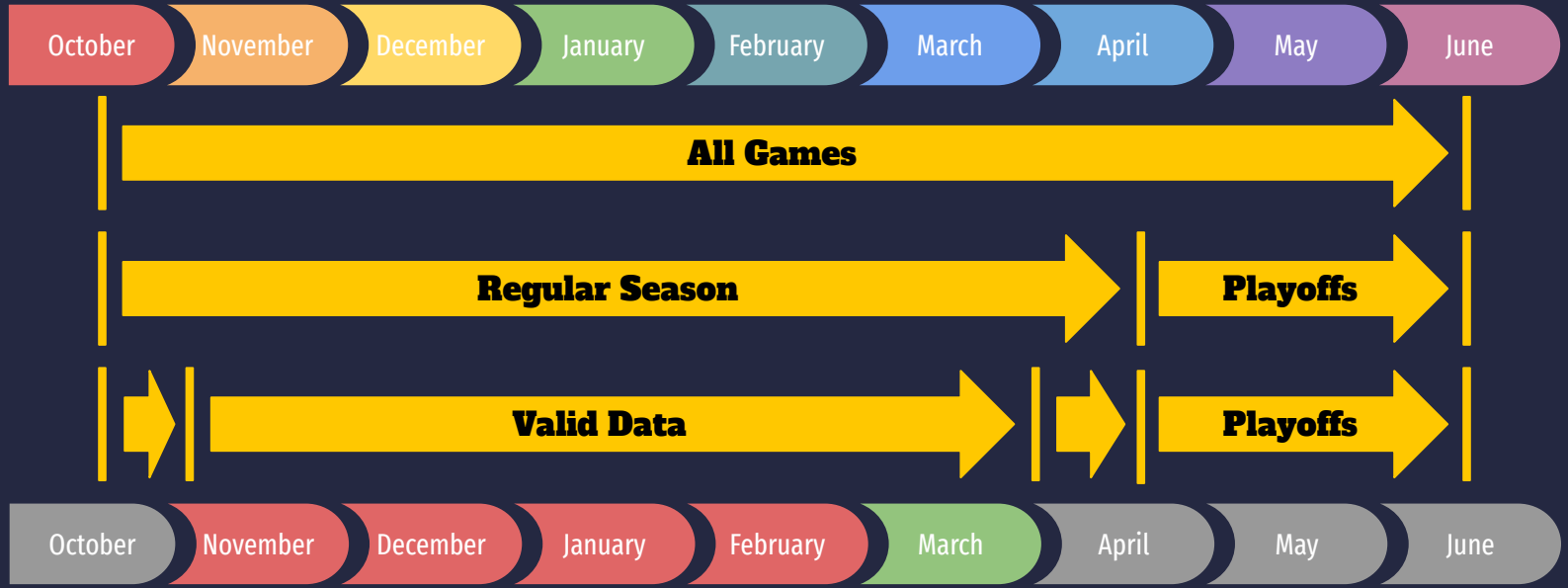
Basketball Season

What Data Are we Looking At?

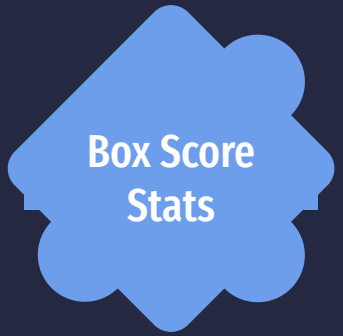
Basketball Season

What Data Are we Looking At?

Basketball Season



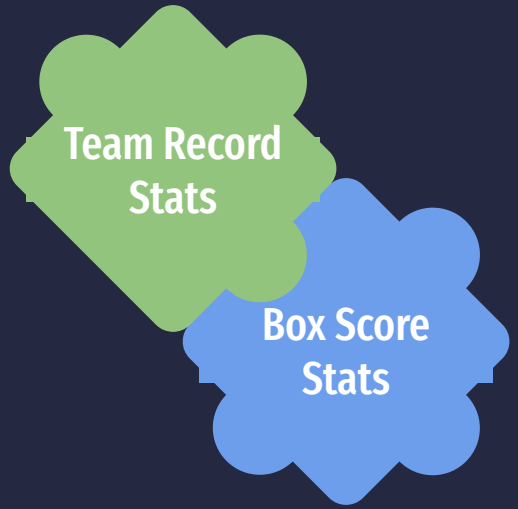
What Data Are we Looking At?



Most Data Comes From Here

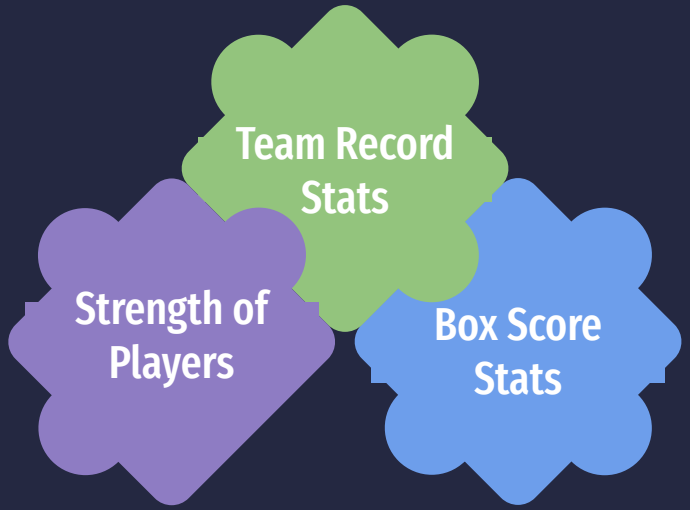
- Average Defense/Offensive Rating
- Average 3P Per Game
- Average FG Per Game
- Average FT Per Game
- Average Points Per Game
  - Total Season
  - Last 5 Games
  - Last 10 Games

What Data Are we Looking At?



- Win Percentage
  - Overall
  - Last 5 Games
  - Last 10 Games
- Home/Away Win Percentage

What Data Are we Looking At?



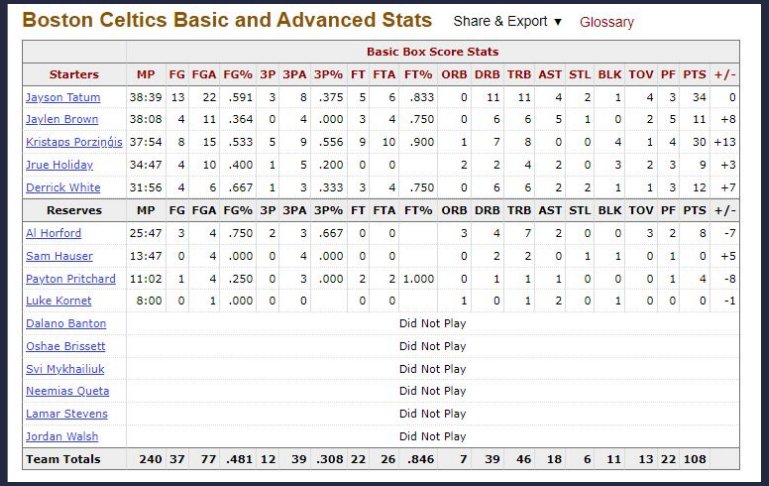
- Strength of Key Players Missing
  - Celtics
  - Opponent

What Data Are we Looking At?

- Rank in Conference
  - Celtics/Opponent
- Rank in Division
  - Celtics/Opponent

What Data Are we Looking At?

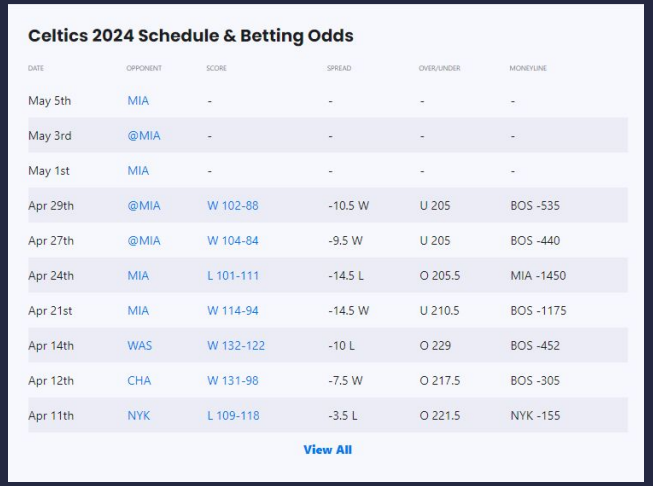
Sources



Boston Celtics Basic and Advanced Stats Share & Export ▾ Glossary

	Basic Box Score Stats																		
Starters	**MP**	**FG**	**FGA**	**FG%**	**3P**	**3PA**	**3P%**	**FT**	**FTA**	**FT%**	**ORB**	**DRB**	**TRB**	**AST**	**STL**	**BLK**	**TOV**	**PF**	**PTS**	**+/-**
Jayson Tatum	38:39	13	22	.591	3	8	.375	5	6	.833	0	11	11	4	2	1	4	3	34	0
Jaylen Brown	38:08	4	11	.364	0	4	.000	3	4	.750	0	6	6	5	1	0	2	5	11	+8
Kristaps Porziņģis	37:54	8	15	.533	5	9	.556	9	10	.900	1	7	8	0	0	4	1	4	30	+13
Jrue Holiday	34:47	4	10	.400	1	5	.200	0	0		2	2	4	2	0	3	2	3	9	+3
Derrick White	31:56	4	6	.667	1	3	.333	3	4	.750	0	6	6	2	2	1	1	3	12	+7
Reserves	**MP**	**FG**	**FGA**	**FG%**	**3P**	**3PA**	**3P%**	**FT**	**FTA**	**FT%**	**ORB**	**DRB**	**TRB**	**AST**	**STL**	**BLK**	**TOV**	**PF**	**PTS**	**+/-**
Al Horford	25:47	3	4	.750	2	3	.667	0	0		3	4	7	2	0	0	3	2	8	-7
Sam Hauser	13:47	0	4	.000	0	4	.000	0	0		0	2	2	0	1	1	0	1	0	+5
Payton Pritchard	11:02	1	4	.250	0	3	.000	2	2	1.000	0	1	1	1	0	0	0	1	4	-8
Luke Kornet	8:00	0	1	.000	0	0		0	0		1	0	1	2	0	1	0	0	0	-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**	

https://www.basketball-reference.com/boxscores/



Celtics 2024 Schedule & Betting Odds

DATE	OPPONENT	SCORE	SPREAD	OVER/UNDER	MONEYLINE
May 5th	MIA	-	-	-	-
May 3rd	@MIA	-	-	-	-
May 1st	MIA	-	-	-	-
Apr 29th	@MIA	W 102-88	-10.5 W	U 205	BOS -535
Apr 27th	@MIA	W 104-84	-9.5 W	U 205	BOS -440
Apr 24th	MIA	L 101-111	-14.5 L	O 205.5	MIA -1450
Apr 21st	MIA	W 114-94	-14.5 W	U 210.5	BOS -1175
Apr 14th	WAS	W 132-122	-10 L	O 229	BOS -452
Apr 12th	CHA	W 131-98	-7.5 W	O 217.5	BOS -305
Apr 11th	NYK	L 109-118	-3.5 L	O 221.5	NYK -155

View All

https://www.actionnetwork.com/nba/odds/boston-celtics

Scoring Metric

```
def custom_profit_score(y, y_pred, celtics_line, celtics_payout, opp_payout, bet=None):
    if bet is None:
        bet = np.ones(len(y))

    if type(bet) in [int, float]:
        bet = np.ones(len(y)) * bet

    bet_on_celtics = y_pred > (celtics_line * -1)
    celtics_win = y > (celtics_line * -1)
    opponent_win = y < (celtics_line * -1)

    payout = ((bet_on_celtics == celtics_win) * (((100/(celtics_payout*-1))*bet*(bet_on_

    return(sum(payout) - sum(bet))
```

Goal: Maximize Profit

Custom CV/Gridsearch

Cross Validation

```
def perform_cross_validation(model, X, y, cv, line, celtics_payout, opp_payout, bet_size):
    scores=[]
    for train_index, test_index in cv.split(X):
        X_training, X_testing = X.iloc[train_index], X.iloc[test_index]
        y_training, y_testing = y.iloc[train_index], y.iloc[test_index]
        line_test = line.iloc[test_index]
        celtics_payout_test =  celtics_payout.iloc[test_index]
        opp_payout_test =  opp_payout.iloc[test_index]

        model_clone = clone(model)
        ypred = model_clone.fit(X_training, y_training).predict(X_testing)
        score = custom_profit_score(y=y_testing.values,
                                    y_pred=ypred,
                                    celtics_line=line_test.values,
                                    celtics_payout=celtics_payout_test.values,
                                    opp_payout=opp_payout_test.values,
                                    bet=bet_size)
        scores.append(score)
    return {'scores': scores}
```

Grid Search

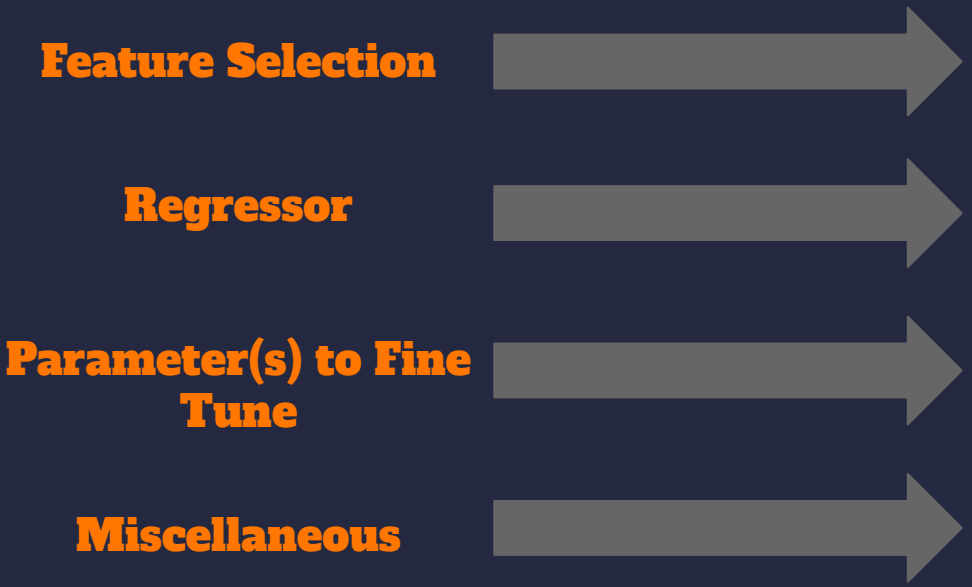
```
def grid_search_custom_cv(model, param_grid, X, y, line, celtics_payout, opp_payout, cv, bet_size):
    results = []

    for params in ParameterGrid(param_grid):
        model_clone = clone(model)
        model_clone.set_params(**params)

        cv_results = perform_cross_validation(model_clone, X, y, cv, line, celtics_payout, opp_payout, bet_size)
        results.append({
            **params,
            'scores': cv_results['scores'],
            'mean_score': np.mean(cv_results['scores']),
            'std_score': np.std(cv_results['scores'])
        })

    return pd.DataFrame(results)
```

Model 1 - Inputs



Model 1 - Results

	y_test	y_pred	celtics_line	celtics_payout	opp_payout	bet	bet_on_celtics	celtics_win	opp_win	payout
0	28	4.421642	-8.5	-110	-110	100.0	0	1	0	0.000000
1	52	-5.056246	-7.5	-110	-110	100.0	0	1	0	0.000000
2	-1	-24.308127	-8.5	-110	-110	100.0	0	0	1	190.909091
3	-6	-21.520836	-2.5	-110	-110	100.0	0	0	1	190.909091
4	10	-3.562045	-5.5	-110	-110	100.0	0	1	0	0.000000
5	22	-19.371579	-11.5	-110	-110	100.0	0	1	0	0.000000
6	16	16.448712	-6.5	-110	-110	100.0	1	1	0	190.909091
7	15	26.960147	-5.5	-110	-110	100.0	1	1	0	190.909091
8	26	67.415050	-14.5	-110	-110	100.0	1	1	0	190.909091
9	25	41.392881	-15.5	-110	-110	100.0	1	1	0	190.909091
10	3	5.786154	-10.5	-110	-110	100.0	0	0	1	190.909091
11	27	45.802205	-13.5	-110	-110	100.0	1	1	0	190.909091
12	11	36.160664	-5.5	-110	-110	100.0	1	1	0	190.909091
13	-2	32.992603	-10.5	-110	-110	100.0	1	0	1	0.000000
14	-1	57.957783	-15.5	-110	-110	100.0	1	0	1	0.000000
15	12	45.560859	-6.5	-110	-110	100.0	1	1	0	190.909091

+$309.09

Profit?????

Model 1 - Results

Highest Coefficients

- Celtics Average Offensive Rating Last 10 Games
- Celtics Average Offensive Rating
- Celtics Average Defensive Rating Last 10 Games
- Celtics Win Percentage Last 5 Games
- Opponent Rank In Conference

Model 2 - Inputs

Model 2 - Results

	y_test	y_pred	celtics_line	celtics_payout	opp_payout	bet	bet_on_celtics	celtics_win	opp_win	payout
0	28	8.083891	-8.5	-110	-110	100.0	0	1	0	0.000000
1	52	12.143246	-7.5	-110	-110	100.0	1	1	0	190.909091
2	-1	32.169300	-8.5	-110	-110	100.0	1	0	1	0.000000
3	-6	8.286247	-2.5	-110	-110	100.0	1	0	1	0.000000
4	10	3.240847	-5.5	-110	-110	100.0	0	1	0	0.000000
5	22	9.998957	-11.5	-110	-110	100.0	0	1	0	0.000000
6	16	0.549139	-6.5	-110	-110	100.0	0	1	0	0.000000
7	15	3.437969	-5.5	-110	-110	100.0	0	1	0	0.000000
8	26	21.792095	-14.5	-110	-110	100.0	1	1	0	190.909091
9	25	18.000477	-15.5	-110	-110	100.0	1	1	0	190.909091
10	3	36.409291	-10.5	-110	-110	100.0	1	0	1	0.000000
11	27	0.525398	-13.5	-110	-110	100.0	0	1	0	0.000000
12	11	16.382109	-5.5	-110	-110	100.0	1	1	0	190.909091
13	-2	38.055009	-10.5	-110	-110	100.0	1	0	1	0.000000
14	-1	25.486353	-15.5	-110	-110	100.0	1	0	1	0.000000
15	12	3.016280	-6.5	-110	-110	100.0	0	1	0	0.000000

-$836.09

No Profit :(

Model 3 - Inputs

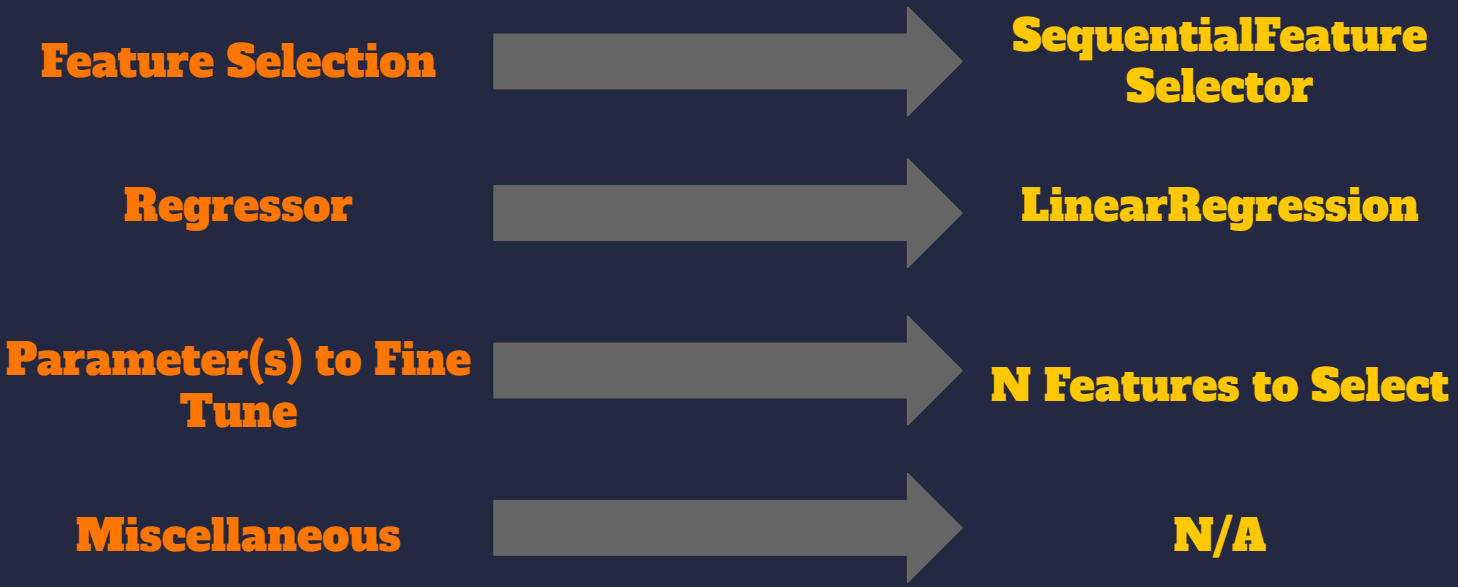
Model 3 - Results

	y_test	y_pred	celtics_line	celtics_payout	opp_payout	bet	bet_on_celtics	celtics_win	opp_win	payout
0	28	8.374595	-8.5	-110	-110	100.0	0	1	0	0.000000
1	52	9.803109	-7.5	-110	-110	100.0	1	1	0	190.909091
2	-1	14.476418	-8.5	-110	-110	100.0	1	0	1	0.000000
3	-6	17.763165	-2.5	-110	-110	100.0	1	0	1	0.000000
4	10	16.459110	-5.5	-110	-110	100.0	1	1	0	190.909091
5	22	16.541414	-11.5	-110	-110	100.0	1	1	0	190.909091
6	16	17.928527	-6.5	-110	-110	100.0	1	1	0	190.909091
7	15	13.871022	-5.5	-110	-110	100.0	1	1	0	190.909091
8	26	24.852029	-14.5	-110	-110	100.0	1	1	0	190.909091
9	25	18.380678	-15.5	-110	-110	100.0	1	1	0	190.909091
10	3	14.464634	-10.5	-110	-110	100.0	1	0	1	0.000000
11	27	24.517520	-13.5	-110	-110	100.0	1	1	0	190.909091
12	11	15.962571	-5.5	-110	-110	100.0	1	1	0	190.909091
13	-2	5.725373	-10.5	-110	-110	100.0	0	0	1	190.909091
14	-1	15.732485	-15.5	-110	-110	100.0	1	0	1	0.000000
15	12	18.100508	-6.5	-110	-110	100.0	1	1	0	190.909091

+$500.00

Profit Again!!

Model 4 - Inputs



Model 4 - Results

	y_test	y_pred	celtics_line	celtics_payout	opp_payout	bet	bet_on_celtics	celtics_win	opp_win	payout
0	28	-14.632646	-8.5	-110	-110	100.0	0	1	0	0.000000
1	52	-11.679664	-7.5	-110	-110	100.0	0	1	0	0.000000
2	-1	24.342374	-8.5	-110	-110	100.0	1	0	1	0.000000
3	-6	24.233088	-2.5	-110	-110	100.0	1	0	1	0.000000
4	10	7.636149	-5.5	-110	-110	100.0	1	1	0	190.909091
5	22	11.153608	-11.5	-110	-110	100.0	0	1	0	0.000000
6	16	24.007052	-6.5	-110	-110	100.0	1	1	0	190.909091
7	15	31.316023	-5.5	-110	-110	100.0	1	1	0	190.909091
8	26	52.553428	-14.5	-110	-110	100.0	1	1	0	190.909091
9	25	31.447832	-15.5	-110	-110	100.0	1	1	0	190.909091
10	3	16.578734	-10.5	-110	-110	100.0	1	0	1	0.000000
11	27	41.407598	-13.5	-110	-110	100.0	1	1	0	190.909091
12	11	-2.301242	-5.5	-110	-110	100.0	0	1	0	0.000000
13	-2	-3.618035	-10.5	-110	-110	100.0	0	0	1	190.909091
14	-1	16.213640	-15.5	-110	-110	100.0	1	0	1	0.000000
15	12	16.807212	-6.5	-110	-110	100.0	1	1	0	190.909091

-$72.72

:(

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Agenda

01

Overview

02

Scraping

03

Machine Learning

04

Takeaways

Learning

- Scraping
- Preprocessing
- Feature Selection
- Custom Scorer
- Custom Cross Validation
- Custom Grid Search

Profitable?

- March Prediction
- 16 Games
- Bet Size: $100/ game
- Profit: $500
- ROI: 31.25%

Website

https://brandon4106.github.io/Fin_377_Swish_Insights/

Thank You!

Questions?