Michael Carvalheiro and Sujay Bhat

12/16/2021

Applied Machine Learning

Final Report

1. Data Collection

We scraped the website https://www.oddsportal.com/ for data on 6226 Premier League matches from the 2005/2006 season to the 2021/2022 season. Our data had the following attributes:

- Season: what season the match was played
- Date: date of the match
- Match_name: which two teams played in the match
- Result: final score of the match
- Home_odd: odds of a home team victory
- Away_odd: odds of an away team victory
- Draw_odd: odds of a draw

2. Data Cleaning

Since we wanted to engineer features for our model, we had to do a bit of data cleaning to the data we scraped off the web:

- Split the result score into home_score and away_score
- Created a "winner" attribute that gives the result of the match, "1" for a home team win, "2" for an away team win, and "0" for a draw. **This would also become our target variable**

- Changing season to be the year the season began (ex. 2015/2016 became 2015)
- Changed the season, home_score, and away_score to integers and home_odd, away_odd, and draw_odd to floats
- Changed the date attribute to datetime type
- Split match_name into home_team and away_team

3. **Feature Engineering**

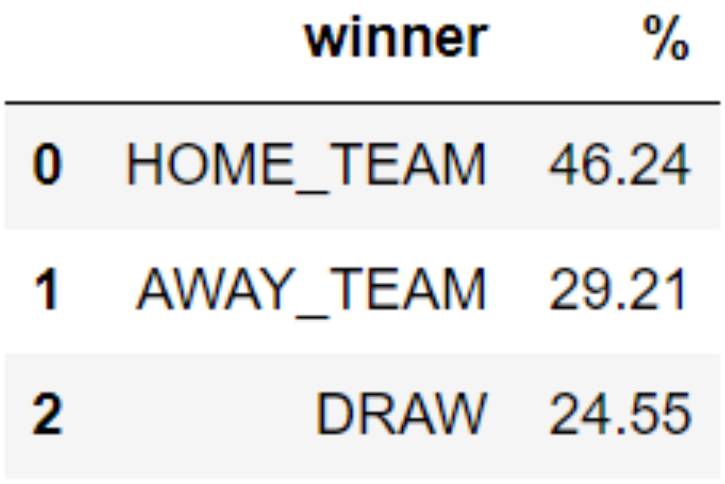
We engineered the following features for both the home team and away team, except for the ls_winner feature. In total, we created 39 features:

- **_rank**: Team's table position in the season so far.
- **_ls_rank**: Team's finishing table position in the previous season.
- **_days_ls_match**: Amount of days since last match.
- **_points**: Team's points in the season so far.
- **_l_points**: Team's average points in the last 3 matches.
- **_goals**: Team's goals in the season so far.
- **_goals_sf**: Team's goals conceded in the season so far.
- **_l_goals**: Team's average goals in the last 3 matches.
- **_l_goals_sf**: Team's average goals conceded in the last 3 matches.
- **_l_wavg_points:** Team's weighted exponential average points in the last 3 games.
- **_l_wavg_goals:** Team's weighted exponential average goals in the last 3 games.
- **_l_wavg_goals_sf**: Team's weighted exponential average goals conceded in the last 3 games.
- **_wins**: Team's wins in the season so far.

- **_draws**: Team's draws in the season so far.
- **_losses**: Team's losses in the season so far.
- **_win_streak**: Team's current win streak.
- **_loss_streak**: Team's current loss streak.
- **_draw_streak:** Team's current draw streak.
- **ls_winner:** Winner of the last match between both teams.

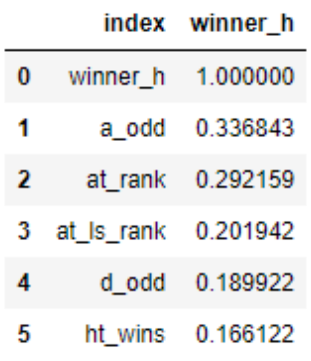
4. Exploratory Data Analysis

The goal here was to perform a brief data exploration in order to better understand the relationships between the dependent and independent variables. The dataset contained 3 possible classes in the target variable, home team winning, away team winning and draw. The classes had the following proportions:

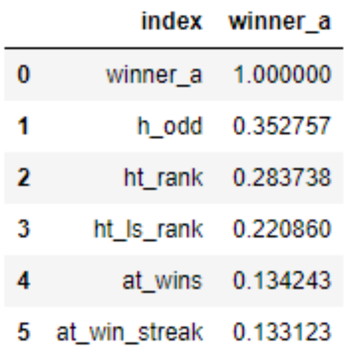


	winner	%
0	HOME_TEAM	46.24
1	AWAY_TEAM	29.21
2	DRAW	24.55

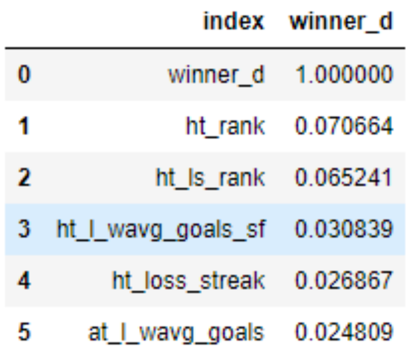
Here are the 5 most correlated features for the home team to win, the away team to win, or a draw:



	index	winner_h
0	winner_h	1.000000
1	a_odd	0.336843
2	at_rank	0.292159
3	at_ls_rank	0.201942
4	d_odd	0.189922
5	ht_wins	0.166122



	index	winner_a
0	winner_a	1.000000
1	h_odd	0.352757
2	ht_rank	0.283738
3	ht_ls_rank	0.220860
4	at_wins	0.134243
5	at_win_streak	0.133123



	index	winner_d
0	winner_d	1.000000
1	ht_rank	0.070664
2	ht_ls_rank	0.065241
3	ht_l_wavg_goals_sf	0.030839
4	ht_loss_streak	0.026867
5	at_l_wavg_goals	0.024809

We also saw that the distributions for the features we engineered were not normally distributed and had different scales, so normalizing the features while constructing our models was necessary.

5. Model Building

We removed all features that wouldn't be available before a match or that wouldn't give us any advantage in terms of predicting the outcome of a match:

```
#dropping columns one wouldn't have before an actual match
cols_to_drop = ['season', 'match_name','date', 'home_team', 'away_team', 'home_score', 'away_score',
                'h_match_points', 'a_match_points']
```

We assigned the "winner" variable to be our target variable:

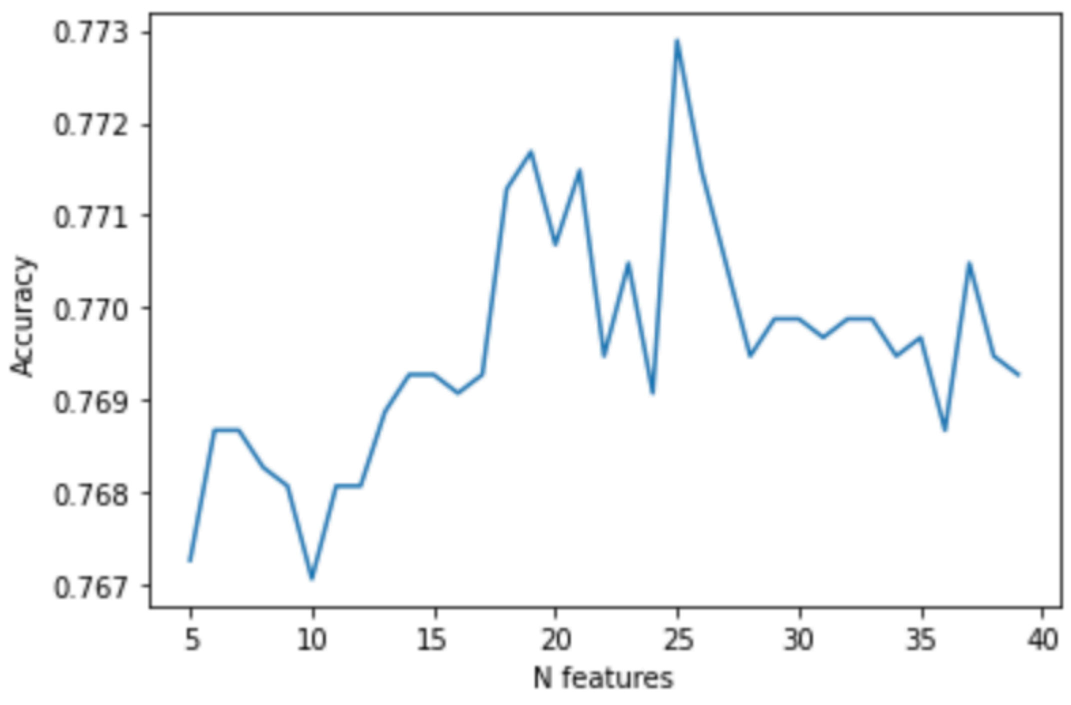
```
#turning the target variable into integers
df['winner'] = np.where(df.winner == 'HOME_TEAM', 1, np.where(df.winner == 'AWAY_TEAM', 2, 0))
```

The accuracies of the 3 models we constructed were 76.9% for the Logistic Regression model, 76.4% for the Random Forest model, and 73.9% for the K-Nearest Neighbors model. All 3 models were accurate, but the Logistic Regression was the winner.

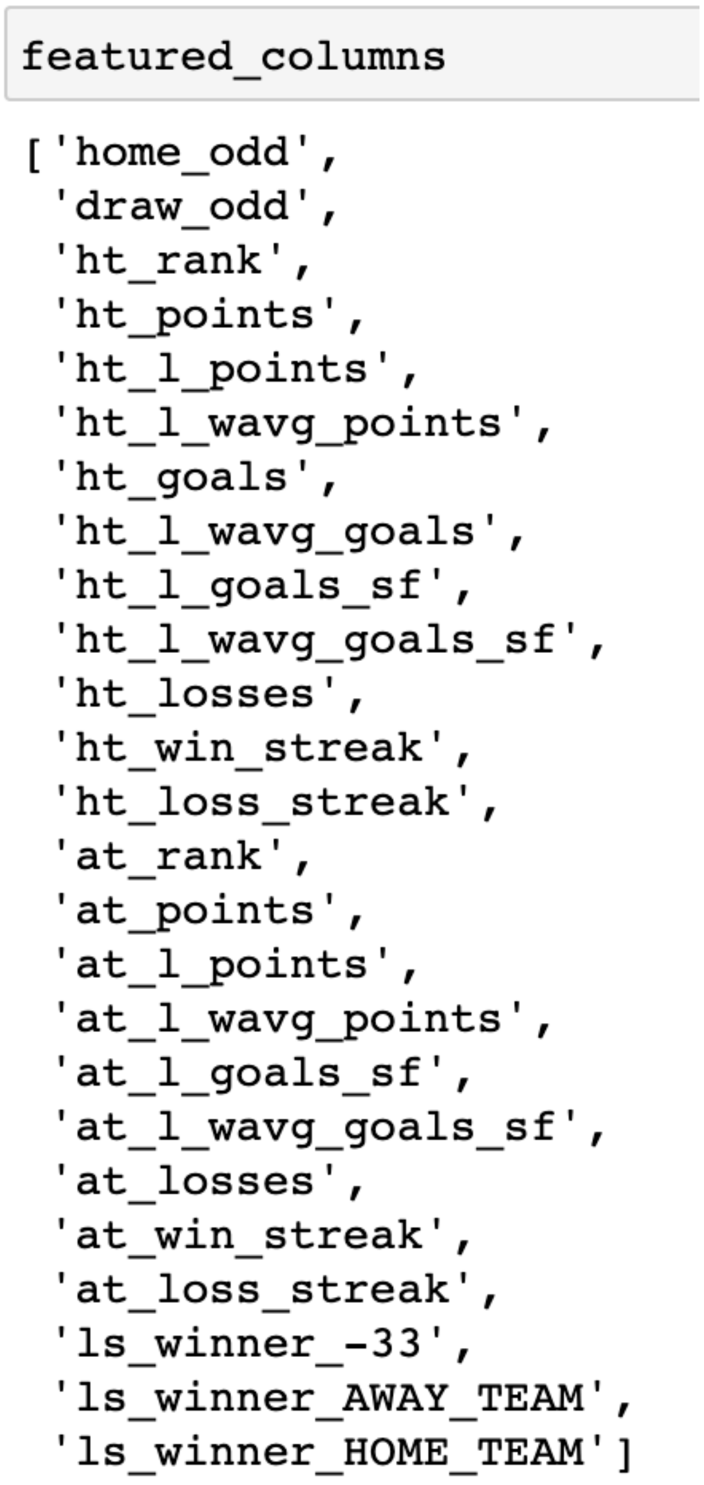
```
Logistic Regression : 0.769, +- 0.004  - Elapsed time:  0.5620009899139404
Random Forest : 0.764, +- 0.007  - Elapsed time:  3.426812171936035
KNN : 0.739, +- 0.005  - Elapsed time:  0.4147980213165283
```

6. Feature Reduction

We wanted to see if we could get the same or an even better accuracy score with less features in the Logistic Regression model. We used the RFE method (recursive feature elimination) to see what the accuracy scores of the Logistic Regression model at each number of features, starting from 5 features and incrementing by 1 all the way up to the max number of features in our original model.



As you can see, the Logistic Regression model had the best accuracy score (around 77.3%) when the number of features was only 25 instead of 39. The 25 features that led to the highest accuracy score in the Logistic Regression model were:



```
featured_columns
```

```
['home_odd',
 'draw_odd',
 'ht_rank',
 'ht_points',
 'ht_l_points',
 'ht_l_wavg_points',
 'ht_goals',
 'ht_l_wavg_goals',
 'ht_l_goals_sf',
 'ht_l_wavg_goals_sf',
 'ht_losses',
 'ht_win_streak',
 'ht_loss_streak',
 'at_rank',
 'at_points',
 'at_l_points',
 'at_l_wavg_points',
 'at_l_goals_sf',
 'at_l_wavg_goals_sf',
 'at_losses',
 'at_win_streak',
 'at_loss_streak',
 'ls_winner_-33',
 'ls_winner_AWAY_TEAM',
 'ls_winner_HOME_TEAM']
```