Using NFL Data to make predictions

A look at how EPA (Estimated Points Added) can be utilized to predict game outcomes.

Play-by-play data for 2022 season as data source

- Python used for general data processing
- ML algorithms used to make predictions
    - Logistic Regression
    - Bagging Classifier

Group 2 - **Michael Morton, Jarrett Lidell, Daniel Boyne, and Severo Fernandez**

nfl-data-py

- nfl_data_py is a Python library for interacting with NFL data sourced from nflfastR, nfldata, dynastyprocess, and Draft Scout.
- Includes import functions for play-by-play data, weekly data, seasonal data, rosters, win totals, scoring lines, officials, draft picks, draft pick values, schedules, team descriptive info, combine results and id mappings across various sites.

Usage

```
import nfl_data_py as nfl
```

Working with play-by-play data

```
nfl.import_pbp_data(years, columns, downcast=True, cache=False, alt_path=None)
```

Returns play-by-play data for the years and columns specified

years : required, list of years to pull data for (earliest available is 1999)

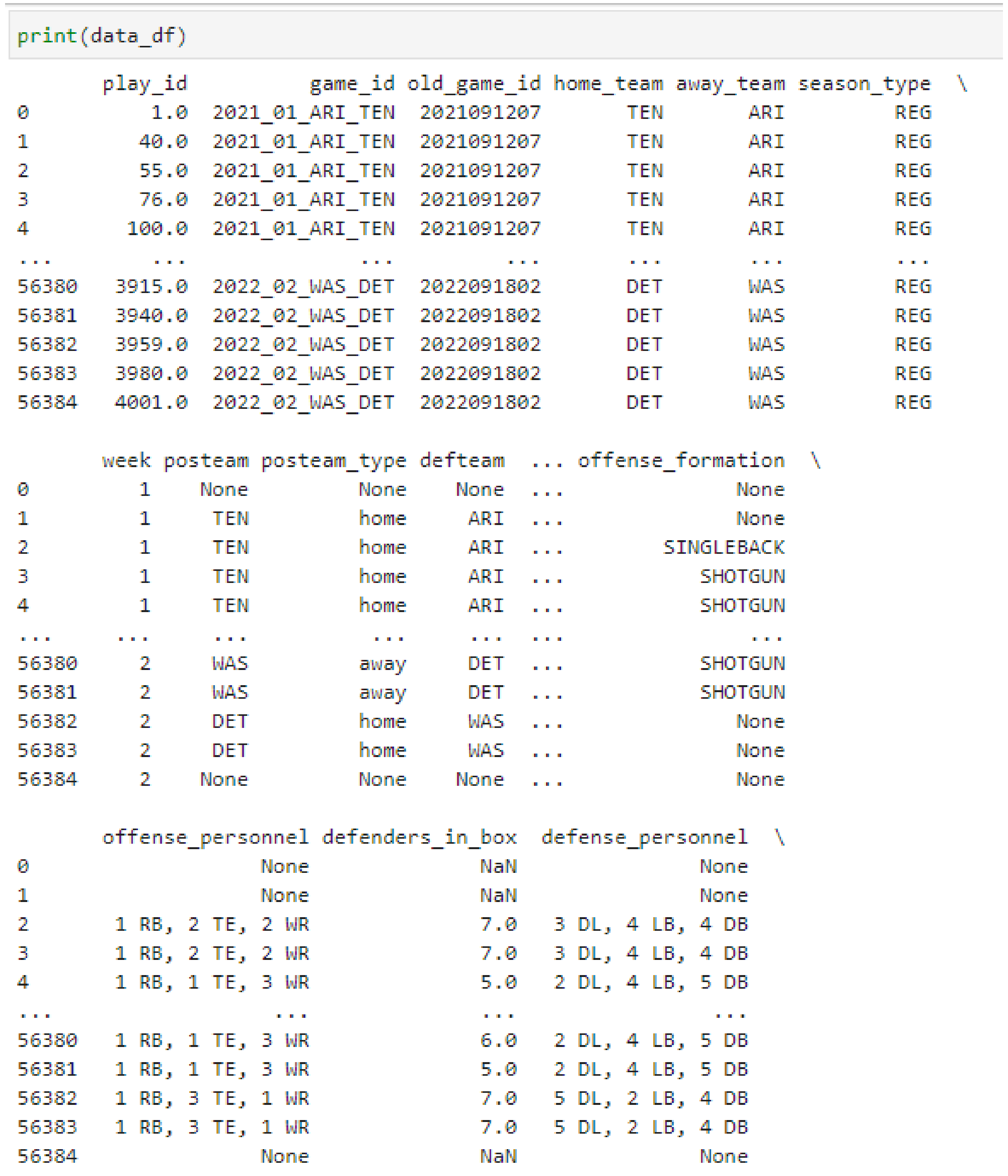
columns : optional, list of columns to pull data for

downcast : optional, converts float64 columns to float32, reducing memory usage by ~30%. Will slow down initial load speed ~50%

cache : optional, determines whether to pull pbp data from github repo or local cache generated by nfl.cache_pbp()

alt_path : optional, required if nfl.cache_pbp() is called using an alternate path to the default cache

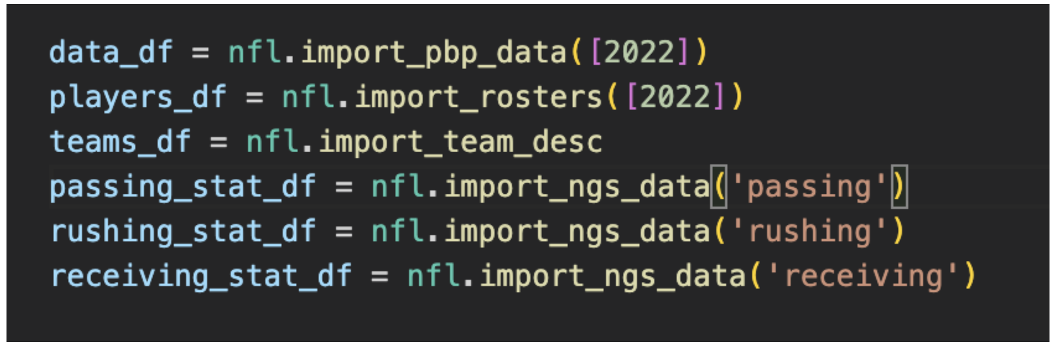
Sample of NFL Play-by-play data



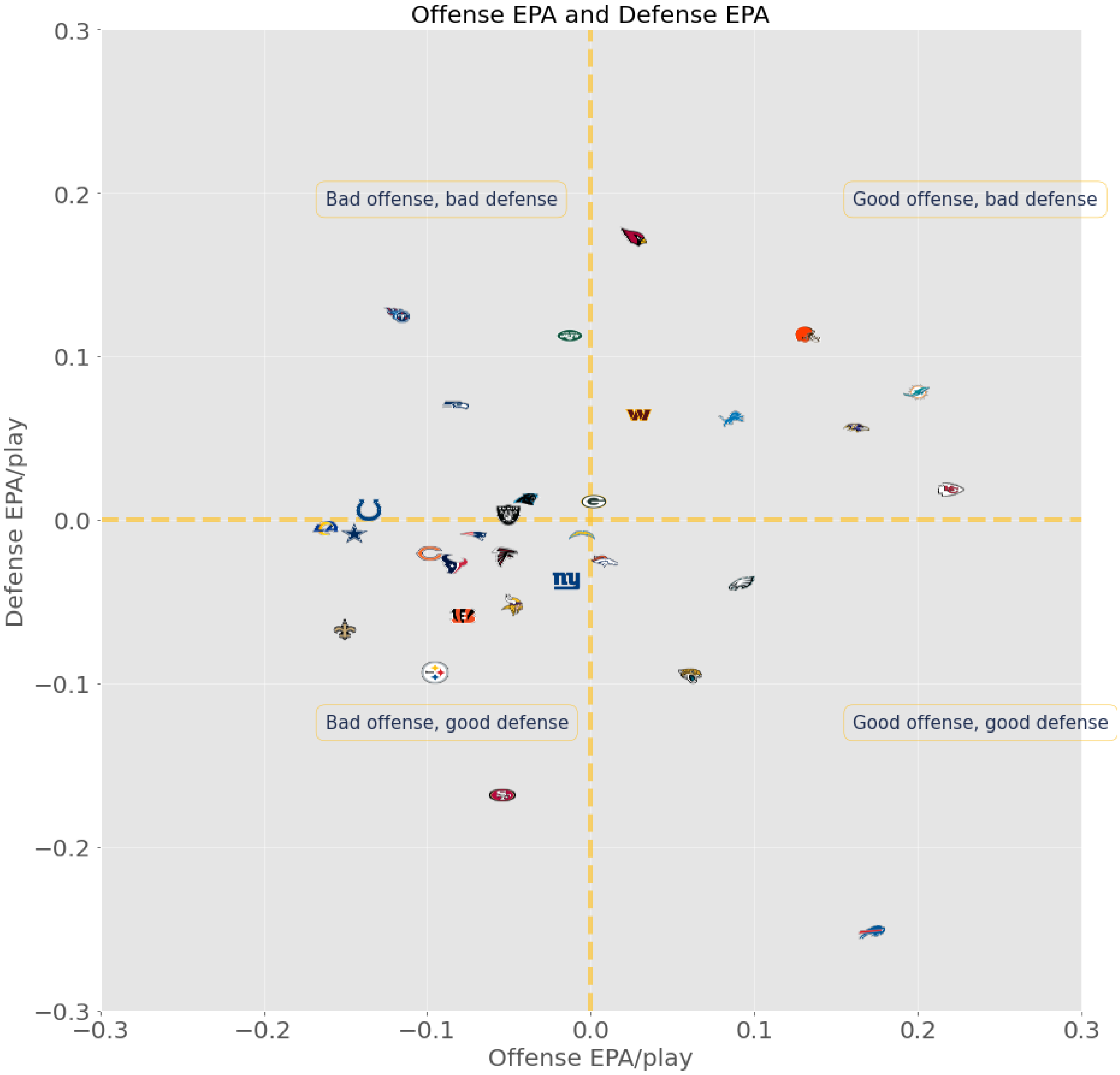
```
print(data_df)
```

```
       play_id          game_id old_game_id home_team away_team season_type  \
0          1.0  2021_01_ARI_TEN  2021091207       TEN       ARI         REG   
1         40.0  2021_01_ARI_TEN  2021091207       TEN       ARI         REG   
2         55.0  2021_01_ARI_TEN  2021091207       TEN       ARI         REG   
3         76.0  2021_01_ARI_TEN  2021091207       TEN       ARI         REG   
4        100.0  2021_01_ARI_TEN  2021091207       TEN       ARI         REG   
...        ...              ...         ...       ...       ...         ...   
56380   3915.0  2022_02_WAS_DET  2022091802       DET       WAS         REG   
56381   3940.0  2022_02_WAS_DET  2022091802       DET       WAS         REG   
56382   3959.0  2022_02_WAS_DET  2022091802       DET       WAS         REG   
56383   3980.0  2022_02_WAS_DET  2022091802       DET       WAS         REG   
56384   4001.0  2022_02_WAS_DET  2022091802       DET       WAS         REG   

       week posteam posteam_type defteam  ... offense_formation  \
0         1    None         None    None  ...              None   
1         1     TEN         home     ARI  ...              None   
2         1     TEN         home     ARI  ...        SINGLEBACK   
3         1     TEN         home     ARI  ...           SHOTGUN   
4         1     TEN         home     ARI  ...           SHOTGUN   
...     ...     ...          ...     ...  ...               ...   
56380     2     WAS         away     DET  ...           SHOTGUN   
56381     2     WAS         away     DET  ...           SHOTGUN   
56382     2     DET         home     WAS  ...              None   
56383     2     DET         home     WAS  ...              None   
56384     2    None         None    None  ...              None   

       offense_personnel defenders_in_box  defense_personnel  \
0                   None              NaN               None   
1                   None              NaN               None   
2       1 RB, 2 TE, 2 WR              7.0   3 DL, 4 LB, 4 DB   
3       1 RB, 2 TE, 2 WR              7.0   3 DL, 4 LB, 4 DB   
4       1 RB, 1 TE, 3 WR              5.0   2 DL, 4 LB, 5 DB   
...                  ...              ...                ...   
56380   1 RB, 1 TE, 3 WR              6.0   2 DL, 4 LB, 5 DB   
56381   1 RB, 1 TE, 3 WR              5.0   2 DL, 4 LB, 5 DB   
56382   1 RB, 3 TE, 1 WR              7.0   5 DL, 2 LB, 4 DB   
56383   1 RB, 3 TE, 1 WR              7.0   5 DL, 2 LB, 4 DB   
56384               None              NaN               None   
```

We also imported the following datasets to establish average EPA/play



```
data_df = nfl.import_pbp_data([2022])
players_df = nfl.import_rosters([2022])
teams_df = nfl.import_team_desc
passing_stat_df = nfl.import_ngs_data('passing')
rushing_stat_df = nfl.import_ngs_data('rushing')
receiving_stat_df = nfl.import_ngs_data('receiving')
```



The model was approximately 65.9% accurate and can definitely be improved. We believe the model may be more useful as a pre-game win probability model.

```
#scaling the data
#A logistic regression model gives nice results with both training and test data

from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X_train)
train_scaled = scaler.transform(X_train)
test_scaled = scaler.transform(X_test)
X_scaled1 = scaler.transform(X)

scaler2 = preprocessing.StandardScaler().fit(X)
X_scaled2 = scaler2.transform(X)
forecast_scaled = scaler2.transform(forecast)
```

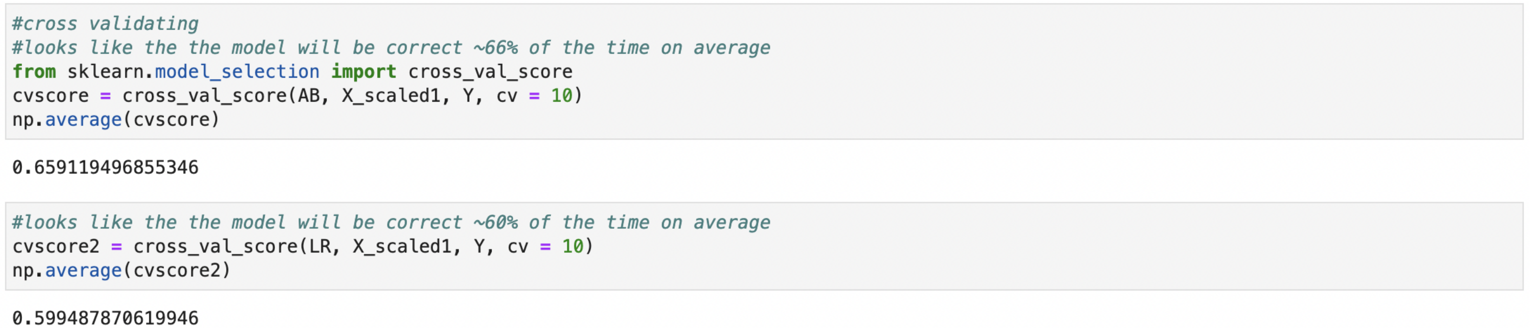
```
AB = BaggingClassifier(base_estimator = SVC(tol = .1, C = .9), n_jobs = -1, n_estimators = 50).fit(train_scaled,Y_train)
print("Test model: ",np.sum(np.array(AB.predict(test_scaled)) == np.array(Y_test))/len(X_test))
print("Train model: ",np.sum(np.array(AB.predict(train_scaled)) == np.array(Y_train))/len(X_train))
```

```
Test model:  0.6287878787878788
Train model:  0.8106060606060606
```

```
LR = LogisticRegression().fit(train_scaled,Y_train)
print("Test model: ", np.sum(np.array(LR.predict(test_scaled)) == np.array(Y_test))/len(X_test))
print("Train model: ",np.sum(np.array(LR.predict(train_scaled)) == np.array(Y_train))/len(X_train))
```

```
Test model:  0.571969696969697
Train model:  0.6515151515151515
```

The first model which ended up having accurate predictions on average 64-66% of the time -

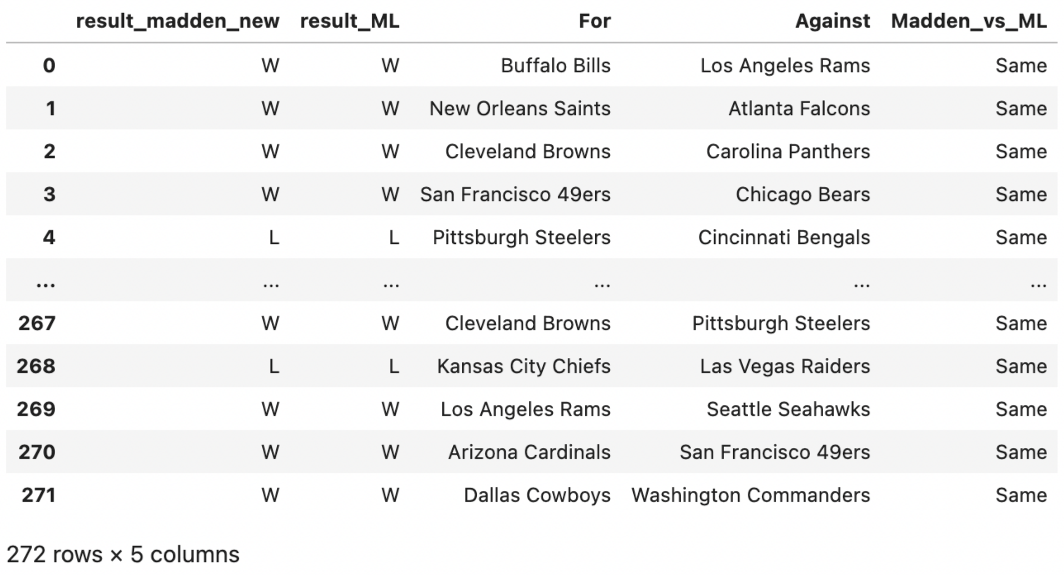


```
#cross validating
#looks like the the model will be correct ~66% of the time on average
from sklearn.model_selection import cross_val_score
cvscore = cross_val_score(AB, X_scaled1, Y, cv = 10)
np.average(cvscore)
```

```
0.659119496855346
```

```
#looks like the the model will be correct ~60% of the time on average
cvscore2 = cross_val_score(LR, X_scaled1, Y, cv = 10)
np.average(cvscore2)
```

```
0.599487870619946
```



	result_madden_new	result_ML	For	Against	Madden_vs_ML
0	W	W	Buffalo Bills	Los Angeles Rams	Same
1	W	W	New Orleans Saints	Atlanta Falcons	Same
2	W	W	Cleveland Browns	Carolina Panthers	Same
3	W	W	San Francisco 49ers	Chicago Bears	Same
4	L	L	Pittsburgh Steelers	Cincinnati Bengals	Same
...	...	...	...	...	...
267	W	W	Cleveland Browns	Pittsburgh Steelers	Same
268	L	L	Kansas City Chiefs	Las Vegas Raiders	Same
269	W	W	Los Angeles Rams	Seattle Seahawks	Same
270	W	W	Arizona Cardinals	San Francisco 49ers	Same
271	W	W	Dallas Cowboys	Washington Commanders	Same

272 rows × 5 columns

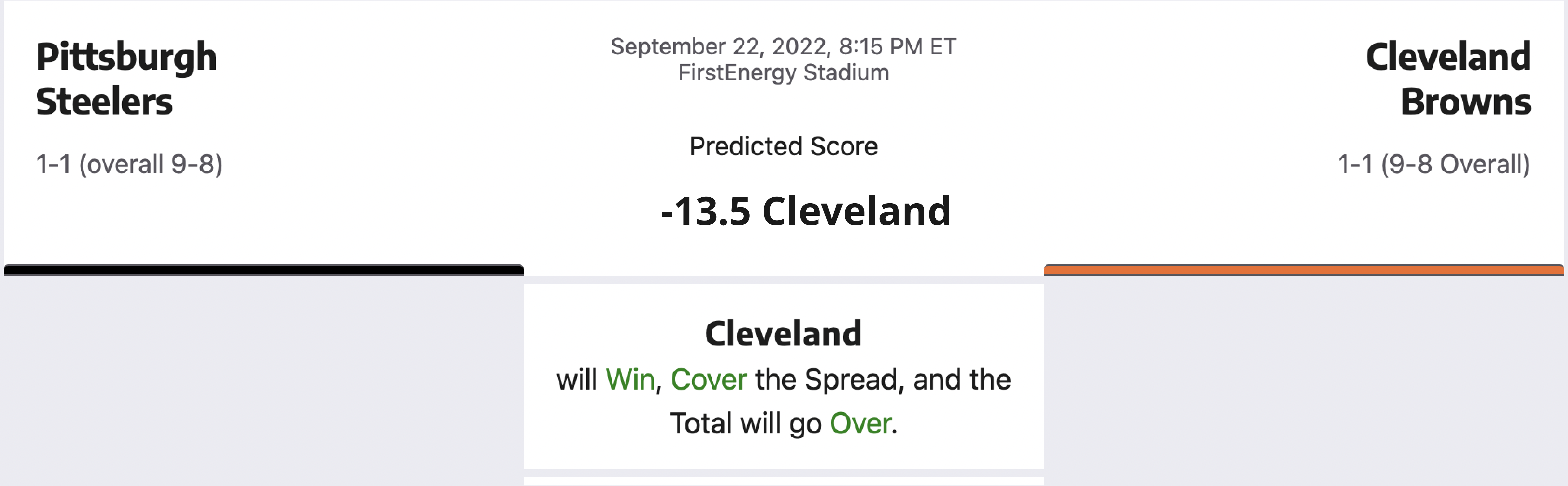
Actual W-L Based on EPA

Week	W-L	Win %
Week 3	0-0	0%
Week 2	9-7	56.3%
Week 1	7-8-1	46.7%
SEASON	**16-15-1**	**51.6%**

What's Next?

We'd like to add EPA adjusted for opponent and also investigate how WP influences EPA predictability, as it's an intriguing topic with potential usefulness. Trying out other elements besides EPA could also be beneficial. Turnover rates, duration of possession, average number of plays run, average starting field position, special teams performance, QB specific play, and so on could all be valuable indicators. We found in the feature importance visualization that passing EPA per play had the strongest predictive power for a home team win and an away team win. Exploring QB specific features could help improve the model. Of course, that would require roster data for each week.

Pittsburgh Steelers vs Cleveland Browns



Pittsburgh Steelers

1-1 (overall 9-8)

September 22, 2022, 8:15 PM ET
FirstEnergy Stadium

Predicted Score

-13.5 Cleveland

Cleveland Browns

1-1 (9-8 Overall)

Cleveland
will Win, Cover the Spread, and the Total will go Over.

Thank you!