NBA Game Predictions

In [143]:

```
import pandas as pd
```

In [144]:

```
df = pd.read_csv("nba_games.csv", index_col=0)
```

In [145]:

```
df
```

Out[145]:

	mp	mp.1	fg	fga	fg%	3p	3pa	3p%	ft	fta	...	tov%_max_opp	usg%_max_opp	ortg_max_opp	drtg_
0	240.0	240.0	39.0	81.0	0.481	6.0	20.0	0.300	14.0	18.0	...	22.8	29.0	178.0	
1	240.0	240.0	36.0	100.0	0.360	7.0	31.0	0.226	16.0	19.0	...	50.0	32.6	152.0	
2	240.0	240.0	37.0	85.0	0.435	8.0	19.0	0.421	17.0	23.0	...	20.0	30.9	148.0	
3	240.0	240.0	41.0	89.0	0.461	8.0	21.0	0.381	17.0	19.0	...	28.6	30.9	138.0	
4	240.0	240.0	27.0	86.0	0.314	6.0	26.0	0.231	15.0	20.0	...	16.8	30.9	157.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	240.0	240.0	35.0	81.0	0.432	11.0	26.0	0.423	27.0	36.0	...	34.2	33.7	160.0	
17768	240.0	240.0	37.0	74.0	0.500	13.0	25.0	0.520	26.0	37.0	...	25.0	30.0	139.0	
17769	240.0	240.0	42.0	89.0	0.472	14.0	33.0	0.424	10.0	20.0	...	25.6	29.9	175.0	
17770	240.0	240.0	41.0	85.0	0.482	9.0	26.0	0.346	26.0	30.0	...	27.7	27.1	150.0	
17771	240.0	240.0	33.0	85.0	0.388	12.0	44.0	0.273	28.0	34.0	...	51.5	36.2	141.0	

17772 rows × 150 columns

In [149]:

```
df = df.sort_values("date")
```

In [150]:

```
df = df.reset_index(drop=True)
```

In [151]:

```
del df["mp.1"]
del df["mp_opp.1"]
del df["index_opp"]
```

In [152]:

```
def add_target(team):
  team["target"] = team["won"].shift(-1)
```

```
    return team

df = df.groupby("team", group_keys=False).apply(add_target)
```

In [153]:

```
df[df["team"] == "PHI"]
```

Out[153]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp	ortg_max_opp	drtg_max_opp	team_op
17	240.0	34.0	83.0	0.410	7.0	22.0	0.318	20.0	23.0	0.870	...	42.9	239.0	107.0	BO
50	240.0	19.0	63.0	0.302	6.0	15.0	0.400	27.0	37.0	0.730	...	33.1	200.0	94.0	UT
101	240.0	38.0	85.0	0.447	7.0	24.0	0.292	17.0	23.0	0.739	...	34.2	171.0	112.0	CL
127	240.0	34.0	81.0	0.420	6.0	28.0	0.214	13.0	15.0	0.867	...	42.0	110.0	105.0	M
167	240.0	41.0	79.0	0.519	10.0	23.0	0.435	10.0	15.0	0.667	...	31.9	177.0	113.0	CL
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17697	240.0	38.0	84.0	0.452	8.0	30.0	0.267	19.0	22.0	0.864	...	33.4	200.0	114.0	M
17702	240.0	32.0	67.0	0.478	16.0	33.0	0.485	19.0	22.0	0.864	...	37.1	121.0	127.0	M
17711	240.0	37.0	68.0	0.544	16.0	33.0	0.485	26.0	34.0	0.765	...	28.6	155.0	129.0	M
17718	240.0	31.0	85.0	0.365	9.0	32.0	0.281	14.0	15.0	0.933	...	34.6	284.0	102.0	M
17725	240.0	36.0	87.0	0.414	12.0	37.0	0.324	6.0	7.0	0.857	...	36.5	152.0	107.0	M

605 rows × 148 columns

In [154]:

```
df["target"][pd.isnull(df["target"])] = 2
df["target"] = df["target"].astype(int, errors="ignore")
```

```
<ipython-input-154-ac49a6f40115>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df["target"][pd.isnull(df["target"])] = 2
```

In [155]:

```
df["won"].value_counts()
```

Out[155]:

```
False    8886
True     8886
Name: won, dtype: int64
```

In [156]:

```
df["target"].value_counts()
```

Out[156]:

```
1    8872
0    8870
```

```
2      30
Name: target, dtype: int64
```

In [157]:

```
nulls = pd.isnull(df)
```

In [158]:

```
nulls = nulls.sum()
```

In [159]:

```
nulls = nulls[nulls > 0]
```

In [160]:

```
valid_columns = df.columns[~df.columns.isin(nulls.index)]
```

In [161]:

```
valid_columns
```

Out[161]:

```
Index(['mp', 'fg', 'fga', 'fg%', '3p', '3pa', '3p%', 'ft', 'fta', 'ft%',
       ...
       'usg%_max_opp', 'ortg_max_opp', 'drtg_max_opp', 'team_opp', 'total_opp',
       'home_opp', 'season', 'date', 'won', 'target'],
      dtype='object', length=142)
```

In [162]:

```
df = df[valid_columns].copy()
```

In [163]:

```
df
```

Out[163]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp	ortg_max_opp	drtg_max_opp	team_op
0	240.0	35.0	83.0	0.422	6.0	18.0	0.333	19.0	27.0	0.704	...	43.7	206.0	104.0	GS
1	240.0	38.0	94.0	0.404	9.0	29.0	0.310	10.0	17.0	0.588	...	34.6	162.0	104.0	C
2	240.0	37.0	87.0	0.425	7.0	19.0	0.368	16.0	23.0	0.696	...	29.0	138.0	105.0	CI
3	240.0	41.0	96.0	0.427	9.0	30.0	0.300	20.0	22.0	0.909	...	38.9	201.0	120.0	NO
4	240.0	37.0	82.0	0.451	8.0	27.0	0.296	12.0	15.0	0.800	...	23.6	132.0	104.0	DE
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	240.0	34.0	85.0	0.400	15.0	38.0	0.395	14.0	19.0	0.737	...	36.3	133.0	112.0	GS
17768	240.0	41.0	88.0	0.466	9.0	40.0	0.225	13.0	15.0	0.867	...	94.4	300.0	112.0	BO
17769	240.0	31.0	75.0	0.413	11.0	32.0	0.344	21.0	31.0	0.677	...	36.2	222.0	107.0	GS
17770	240.0	34.0	80.0	0.425	11.0	28.0	0.393	11.0	12.0	0.917	...	31.5	186.0	111.0	GS
17771	240.0	38.0	92.0	0.413	19.0	46.0	0.413	8.0	8.0	1.000	...	42.6	141.0	126.0	BO

17772 rows × 142 columns

In [164]:

```
from sklearn.model_selection import TimeSeriesSplit
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import RidgeClassifier

rr = RidgeClassifier(alpha=1)
split = TimeSeriesSplit(n_splits=3)

sfs = SequentialFeatureSelector(rr,
                                n_features_to_select=30,
                                direction="forward",
                                cv=split,
                                n_jobs=1
                               )
```

In [165]:

```
removed_columns = ["season", "date", "won", "target", "team", "team_opp"]
selected_columns = df.columns[~df.columns.isin(removed_columns)]
```

In [167]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df[selected_columns] = scaler.fit_transform(df[selected_columns])
```

In [168]:

```
df
```

Out[168]:

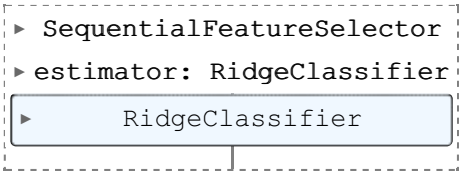
	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp	ort
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.277279	
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.160462	
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.088575	
3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.215661	
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.019255	
...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.182285	
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.928113	
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.181001	
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.120668	
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.263158	

17772 rows × 142 columns

In [169]:

```
sfs.fit(df[selected_columns], df["target"])
```

Out[169]:



In [170]:

```
predictors = list(selected_columns[sfs.get_support()])
```

In [171]:

```
predictors
```

Out[171]:

```
['mp',
 'fg%',
 'orb',
 'usg%',
 '3p%_max',
 'tov_max',
 '+/-_max',
 'efg%_max',
 'drb%_max',
 'trb%_max',
 'tov%_max',
 'usg%_max',
 'ortg_max',
 'home',
 'mp_opp',
 '3p_opp',
 'blk_opp',
 'ftr_opp',
 'usg%_opp',
 'fga_max_opp',
 '3p%_max_opp',
 'ft_max_opp',
 'blk_max_opp',
 'pf_max_opp',
 'pts_max_opp',
 'ftr_max_opp',
 'drb%_max_opp',
 'stl%_max_opp',
 'blk%_max_opp',
 'home_opp']
```

In [172]:

```
def backtest(data, model, predictors, start=2, step=1):
    all_predictions = []

    seasons = sorted(data["season"].unique())

    for i in range(start, len(seasons), step):
        season = seasons[i]
        train = data[data["season"] < season]
        test = data[data["season"] == season]

        model.fit(train[predictors], train["target"])

        preds = model.predict(test[predictors])
        preds = pd.Series(preds, index=test.index)
        combined = pd.concat([test["target"], preds], axis=1)
        combined.columns = ["actual", "prediction"]

        all_predictions.append(combined)
```

```
    return pd.concat(all_predictions)
```

In [173]:

```
predictions = backtest(df, rr, predictors)
```

In [174]:

```
predictions
```

Out[174]:

	actual	prediction
5250	1	1
5251	1	1
5252	0	0
5253	1	0
5254	0	1
...	...	...
17767	0	1
17768	1	1
17769	0	1
17770	2	1
17771	2	1

12522 rows × 2 columns

In [175]:

```
from sklearn.metrics import accuracy_score

predictions = predictions[predictions["actual"] != 2]
accuracy_score(predictions["actual"], predictions["prediction"])
```

Out[175]:

```
0.5401857188600705
```

In [176]:

```
df.groupby("home").apply(lambda x: x[x["won"] == 1].shape[0] / x.shape[0])
```

Out[176]:

```
home
0.0    0.428314
1.0    0.571686
dtype: float64
```

In [177]:

```
df
```

Out[177]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp	ort
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.277279	
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.160462	
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.088575	

3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	usg%_max_opp	ort
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.019255	
...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.182285	
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.928113	
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.181001	
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.120668	
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.263158	

17772 rows × 142 columns

In [178]:

```
df_rolling = df[list(selected_columns) +["won", "team", "season"] ]
```

In [179]:

```
df_rolling
```

Out[179]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	blk%_max_opp	tov
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.079	
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.140	
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.185	
3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.063	
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.047	
...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.103	
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.124	
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.076	
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.063	
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.160	

17772 rows × 139 columns

In [180]:

```
df_rolling = df[list(selected_columns) +["won", "team", "season"] ]
def find_team_averages(team):
  rolling = team.rolling(10).mean()
  return rolling

  df_rolling = df_rolling.groupby(["team", "season"], groups_keys=False.apply(find_teams
_averages))
```

In [181]:

```
df_rolling
```

Out[181]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	blk%_max_opp	tov
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.079	
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.140	
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.185	
3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.063	
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.047	
...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.103	
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.124	
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.076	
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.063	
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.160	

17772 rows × 139 columns

In [182]:

```
rolling_cols = [f"{col}_10" for col in df_rolling.columns]
df_rolling.columns = rolling_cols

df = pd.concat([df,df_rolling], axis=1)
```

In [184]:

```
df = df.dropna()
```

In [185]:

```
df
```

Out[185]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	blk%_max_opp_10
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.079
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.140
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.185
3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.063
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.047
...	...	...	...	...	...	...	...	...	...	...	...	...
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.103
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.124
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.076
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.063
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.160

17772 rows × 281 columns

In [186]:

```
def shift_col(team, col_name):
  next_col = team[col_name].shift(-1)
  return next_col
```

```
def add_col(df,col_name):
  return df.groupby("team", group_keys=False).apply(lambda x: shift_col(x, col_name))

df["home_next"] = add_col(df, "home")
df["team_opp_next"] = add_col(df, "team_opp")
df["date_next"] = add_col(df, "date")
```

In [187]:

```
df
```

Out[187]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	ortg_max_opp_10	d
0	0.0	0.363636	0.338235	0.366029	0.206897	0.212121	0.395487	0.418605	0.412698	0.654609	...	0.554502	
1	0.0	0.431818	0.500000	0.322967	0.310345	0.378788	0.368171	0.209302	0.253968	0.519253	...	0.345972	
2	0.0	0.409091	0.397059	0.373206	0.241379	0.227273	0.437055	0.348837	0.349206	0.645274	...	0.232227	
3	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.530806	
4	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.203791	
...	...	...	...	...	...	...	...	...	...	...	...	...	
17767	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.208531	
17768	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	1.000000	
17769	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.630332	
17770	0.0	0.340909	0.294118	0.373206	0.379310	0.363636	0.466746	0.232558	0.174603	0.903151	...	0.459716	
17771	0.0	0.431818	0.470588	0.344498	0.655172	0.636364	0.490499	0.162791	0.111111	1.000000	...	0.246445	

17772 rows × 284 columns

In [188]:

```
full = df.merge(
    df[rolling_cols + ["team_opp_next", "date_next", "team"]],
    left_on=["team", "date_next"],
    right_on=["team_opp_next", "date_next"]
    )
```

In [189]:

```
full
```

Out[189]:

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp_10_
0	0.0	0.500000	0.529412	0.377990	0.310345	0.393939	0.356295	0.441860	0.333333	0.893816	...	0.10397
1	0.0	0.409091	0.323529	0.435407	0.275862	0.348485	0.351544	0.255814	0.222222	0.766628	...	0.13350
2	0.0	0.568182	0.411765	0.552632	0.413793	0.424242	0.445368	0.255814	0.206349	0.833139	...	0.10012
3	0.0	0.500000	0.352941	0.523923	0.448276	0.378788	0.532067	0.232558	0.253968	0.588098	...	0.15789
4	0.0	0.477273	0.367647	0.483254	0.344828	0.257576	0.565321	0.465116	0.476190	0.623104	...	0.22721
...	...	...	...	...	...	...	...	...	...	...	...	...
17733	0.0	0.545455	0.426471	0.511962	0.448276	0.469697	0.440618	0.372093	0.365079	0.659277	...	0.08600

	mp	fg	fga	fg%	3p	3pa	3p%	ft	fta	ft%	...	usg%_max_opp_10_
17734	0.0	0.477273	0.455882	0.409091	0.517241	0.590909	0.414489	0.255814	0.222222	0.766628	...	0.18228
17735	0.0	0.340909	0.367647	0.313397	0.517241	0.515152	0.469121	0.302326	0.285714	0.693116	...	0.13222
17736	0.0	0.500000	0.411765	0.471292	0.310345	0.545455	0.267221	0.279070	0.222222	0.844807	...	0.18100
17737	0.0	0.272727	0.220588	0.344498	0.379310	0.424242	0.408551	0.465116	0.476190	0.623104	...	0.92811

17738 rows × 425 columns

In [190]:

```
full[["team_x", "team_opp_next_x", "team_y", "team_opp_next_y", "date_next"]]
```

Out[190]:

	team_x	team_opp_next_x	team_y	team_opp_next_y	date_next
0	GSW	HOU	HOU	GSW	2015-10-30
1	ATL	NYK	NYK	ATL	2015-10-29
2	POR	PHO	PHO	POR	2015-10-30
3	CLE	MIA	MIA	CLE	2015-10-30
4	DAL	LAC	LAC	DAL	2015-10-29
...	...	...	...	...	...
17733	BOS	GSW	GSW	BOS	2022-06-10
17734	GSW	BOS	BOS	GSW	2022-06-13
17735	BOS	GSW	GSW	BOS	2022-06-13
17736	GSW	BOS	BOS	GSW	2022-06-16
17737	BOS	GSW	GSW	BOS	2022-06-16

17738 rows × 5 columns

In [191]:

```
removed_columns = list(full.columns[full.dtypes == "object"]) + removed_columns
```

In [192]:

```
removed_columns
```

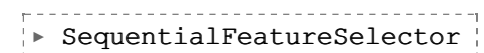
Out[192]:

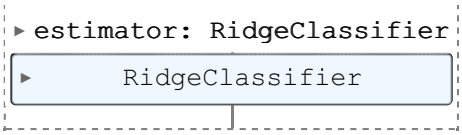
```
['team_x',
 'team_opp',
 'date',
 'team_10_x',
 'team_opp_next_x',
 'date_next',
 'team_10_y',
 'team_opp_next_y',
 'team_y',
 'season',
 'date',
 'won',
 'target',
 'team',
 'team_opp']
```

In [193]:

```
selected_columns = full.columns[~full.columns.isin(removed_columns)]
sfs.fit(full[selected_columns], full["target"])
```

Out[193]:





In [194]:

```
predictors = list(selected_columns[sfs.get_support()])
```

In [195]:

```
predictors
```

Out[195]:

```
['fga',
 'orb%',
 'usg%',
 '+/-_max',
 'drb%_max',
 'usg%_max',
 '3p%_opp',
 'blk_opp',
 'usg%_opp',
 '3p_max_opp',
 'pf_max_opp',
 'ortg_max_opp',
 'usg%_10_x',
 '+/-_max_10_x',
 '3p%_opp_10_x',
 'usg%_opp_10_x',
 'home_next',
 'mp_10_y',
 'fga_10_y',
 'fta_10_y',
 'usg%_10_y',
 'ft_max_10_y',
 'pf_max_10_y',
 '+/-_max_10_y',
 'trb%_max_10_y',
 'usg%_opp_10_y',
 'drb%_max_opp_10_y',
 'tov%_max_opp_10_y',
 'ortg_max_opp_10_y',
 'season_10_y']
```

In [196]:

```
predictions = backtest(full, rr, predictors)
```

In [197]:

```
accuracy_score(predictions["actual"], predictions["prediction"])
```

Out[197]:

```
0.5802113352545629
```