In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('player_mvp_stats.csv')
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	Rk	Player	Pos	Age	Team	G	GS	MP	FG	...	Pts Won	Pts Max	Share	W	L	W/L%	GB	PS/G	PA/G
0	0	135.0	A.C. Green	PF	27.0	LAL	82.0	21.0	26.4	3.1	...	0.0	0.0	0.0	58.0	24.0	0.707	5.0	106.3	99.6
1	1	301.0	Byron Scott	SG	29.0	LAL	82.0	82.0	32.1	6.1	...	0.0	0.0	0.0	58.0	24.0	0.707	5.0	106.3	99.6
2	2	51.0	Elden Campbell	PF	22.0	LAL	52.0	0.0	7.3	1.1	...	0.0	0.0	0.0	58.0	24.0	0.707	5.0	106.3	99.6
3	3	330.0	Irving Thomas	PF	25.0	LAL	26.0	0.0	4.2	0.7	...	0.0	0.0	0.0	58.0	24.0	0.707	5.0	106.3	99.6
4	4	384.0	James Worthy	SF	29.0	LAL	78.0	74.0	38.6	9.2	...	0.0	0.0	0.0	58.0	24.0	0.707	5.0	106.3	99.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16389	16389	330.0	Jordan McLaughlin	PG	23.0	MIN	30.0	2.0	19.7	2.9	...	0.0	0.0	0.0	19.0	45.0	0.297	22.5	113.3	117.5
16390	16390	381.0	Josh Okogie	SG	21.0	MIN	62.0	28.0	25.0	2.7	...	0.0	0.0	0.0	19.0	45.0	0.297	22.5	113.3	117.5
16391	16391	477.0	Karl-Anthony Towns	C	24.0	MIN	35.0	35.0	33.9	9.0	...	0.0	0.0	0.0	19.0	45.0	0.297	22.5	113.3	117.5
16392	16392	317.0	Kelan Martin	SF	24.0	MIN	31.0	4.0	16.0	2.3	...	0.0	0.0	0.0	19.0	45.0	0.297	22.5	113.3	117.5
16393	16393	420.0	Naz Reid	C	20.0	MIN	30.0	11.0	16.5	3.3	...	0.0	0.0	0.0	19.0	45.0	0.297	22.5	113.3	117.5

16394 rows × 42 columns

In [4]:
```
# to do data cleaning before ml
del df['Unnamed: 0']
```

In [232]:
```
df.isna().sum().head(2)
```

Out[232]:
```
Rk        0
Player    0
dtype: int64
```

In [18]:
```
df[df['3P%'].isna()][['3PA','3P']].value_counts()
```

Out[18]:
```
3PA  3P
0.0  0.0    2397
dtype: int64
```

- all players with missing 3P% have zero attemps

In [19]:
```
df = df.fillna(0)
```

In [233]:
```
df.isna().sum().head(2)
```

Out[233]:
```
Rk        0
Player    0
dtype: int64
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['Rk', 'Player', 'Pos', 'Age', 'Team', 'G', 'GS', 'MP', 'FG', 'FGA',
       'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'Year', 'Pts Won', 'Pts Max', 'Share', 'W', 'L', 'W/L%', 'GB', 'PS/G',
       'PA/G', 'SRS'],
      dtype='object')
```

In [24]:
```
# get ride of columns that directly affect the y, causing overfit problem
predictors = ['Age','G', 'GS', 'MP', 'FG', 'FGA',
       'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'Year','W', 'L', 'W/L%', 'GB', 'PS/G',
       'PA/G', 'SRS']
```

In [25]:
```
train = df.query('Year < 2021')
```

In [27]:
```
test = df.query('Year == 2021')
```

In [29]:
```
from sklearn.linear_model import Ridge # ridge prevent overfit
```

In [31]:
```
reg = Ridge(alpha=0.1)
```

In [47]:
```
reg.fit(train[predictors],train['Share'])
```

Out[47]:
```
Ridge
Ridge(alpha=0.1)
```

In [50]:
```
test.shape
```

Out[50]:
```
(619, 41)
```

In [51]:
```
prediction_1s = reg.predict(test[predictors])
```

In [52]:
```
pd_pred = pd.DataFrame(prediction_1s,columns=['prediction'],index = test.index)
```

In [53]:
```
pd_pred
```

Out[53]:

	prediction
1175	0.014023
1176	0.009959
1177	-0.013762
1178	-0.017826
1179	0.007912
...	...
16117	-0.008192
16118	-0.000126
16119	-0.003047
16120	-0.023829
16121	-0.006096

619 rows × 1 columns

In [68]:
```
combination = pd.concat([pd_pred,test[['Share','Player']]],axis=1)
```

In [69]:
```
combination
```

Out[69]:

	prediction	Share	Player
1175	0.014023	0.0	Aaron Gordon
1176	0.009959	0.0	Aaron Gordon
1177	-0.013762	0.0	Al-Farouq Aminu
1178	-0.017826	0.0	Al-Farouq Aminu
1179	0.007912	0.0	Alex Len
...	...	...	...
16117	-0.008192	0.0	Rodney McGruder
16118	-0.000126	0.0	Saben Lee
16119	-0.003047	0.0	Saddiq Bey
16120	-0.023829	0.0	Sekou Doumbouya
16121	-0.006096	0.0	Wayne Ellington

619 rows × 3 columns

In [75]:
```
combination.sort_values('Share',ascending=False).head(10)
```

Out[75]:

	prediction	Share	Player
13532	0.145136	0.961	Nikola Jokić
10473	0.156836	0.580	Joel Embiid
5089	0.139247	0.449	Stephen Curry
12018	0.200518	0.345	Giannis Antetokounmpo
2450	0.068735	0.138	Chris Paul
13156	0.145498	0.042	Luka Dončić
9208	0.110186	0.038	Damian Lillard
4983	0.083648	0.020	Julius Randle
1223	0.028737	0.010	Derrick Rose
1224	0.024673	0.010	Derrick Rose

- how to improve model: identify an error metric

In [77]:
```
from sklearn.metrics import mean_squared_error

mean_squared_error(test['Share'],prediction_1s)
```

Out[77]:
```
0.0023789454142902886
```

In [79]:
```
combination = combination.sort_values("Share",ascending=False)
combination['Rk'] = list(range(1,combination.shape[0]+1))
```

In [80]:
```
combination
```

Out[80]:

	prediction	Share	Player	Rk
13532	0.145136	0.961	Nikola Jokić	1
10473	0.156836	0.580	Joel Embiid	2
5089	0.139247	0.449	Stephen Curry	3
12018	0.200518	0.345	Giannis Antetokounmpo	4
2450	0.068735	0.138	Chris Paul	5
...	...	...	...	...
2720	-0.000281	0.000	Devon Dotson	615
2721	-0.031390	0.000	Garrett Temple	616
2722	0.018648	0.000	Lauri Markkanen	617
2723	-0.019941	0.000	Patrick Williams	618
16121	-0.006096	0.000	Wayne Ellington	619

619 rows × 4 columns

In [82]:
```
combination = combination.sort_values('prediction',ascending=False)
```

In [83]:
```
combination['Predict_Rk'] = list(range(1,combination.shape[0]+1))
```

In [84]:
```
combination
```

Out[84]:

	prediction	Share	Player	Rk	Predict_Rk
12018	0.200518	0.345	Giannis Antetokounmpo	4	1
10473	0.156836	0.580	Joel Embiid	2	2
13156	0.145498	0.042	Luka Dončić	6	3
13532	0.145136	0.961	Nikola Jokić	1	4
5162	0.140184	0.001	LeBron James	17	5
...	...	...	...	...	...
1296	-0.047716	0.000	P.J. Tucker	376	615
16115	-0.048117	0.000	Killian Hayes	212	616
11457	-0.048408	0.000	Patrick McCaw	77	617
9669	-0.052177	0.000	Anžejs Pasečņiks	126	618
13928	-0.053230	0.000	Didi Louzada	186	619

619 rows × 5 columns

In [86]:
```
combination.sort_values('Share',ascending=False).head(10)
```

Out[86]:

	prediction	Share	Player	Rk	Predict_Rk
13532	0.145136	0.961	Nikola Jokić	1	4
10473	0.156836	0.580	Joel Embiid	2	2
5089	0.139247	0.449	Stephen Curry	3	6
12018	0.200518	0.345	Giannis Antetokounmpo	4	1
2450	0.068735	0.138	Chris Paul	5	36
13156	0.145498	0.042	Luka Dončić	6	3
9208	0.110186	0.038	Damian Lillard	7	13
4983	0.083648	0.020	Julius Randle	8	28
1224	0.024673	0.010	Derrick Rose	10	108
1223	0.028737	0.010	Derrick Rose	9	92

- to see how long to take to include that person

In [95]:
```
def find_ap(combination):
    actual = combination.sort_values('Share',ascending=False).head(5)
    predicted = combination.sort_values('prediction',ascending=False)
    ps = []
    found = 0
    seen = 1
    for index, row in predicted.iterrows():
        if row['Player'] in actual['Player'].values:
            found +=1
            ps.append(found/seen)
        seen += 1
    return sum(ps)/len(ps)
```

In [96]:
```
find_ap(combination)
```

Out[96]:
```
0.711111111111111
```

In [94]:
```
x
```

Out[94]:

	prediction	Share	Player	Rk	Predict_Rk
12018	0.200518	0.345	Giannis Antetokounmpo	4	1
10473	0.156836	0.580	Joel Embiid	2	2
13156	0.145498	0.042	Luka Dončić	6	3
13532	0.145136	0.961	Nikola Jokić	1	4
5162	0.140184	0.001	LeBron James	17	5
...	...	...	...	...	...
1296	-0.047716	0.000	P.J. Tucker	376	615
16115	-0.048117	0.000	Killian Hayes	212	616
11457	-0.048408	0.000	Patrick McCaw	77	617
9669	-0.052177	0.000	Anžejs Pasečņiks	126	618
13928	-0.053230	0.000	Didi Louzada	186	619

619 rows × 5 columns

In [97]:
```
years = list(range(1991,2022))
```

In [111]:
```
aps = []
all_prediction = []
for year in years[5:]:
    train = df[df['Year'] < year]
    test = df[df['Year'] == year]
    reg.fit(train[predictors],train['Share'])
    predictions = reg.predict(test[predictors])
    predictions = pd.DataFrame(predictions,columns=['prediction'],index = test.index)
    combination = pd.concat([test[['Share','Player']],predictions],axis=1)
    all_prediction.append(combination)
    aps.append(find_ap(combination))
```

In [123]:
```
all_prediction
```

```
        Share           Player  prediction
513       0.0      A.J. Guyton   -0.007910
514       0.0   Charles Oakley   -0.024502
515       0.0  Dalibor Bagarić   -0.004729
516       0.0   Eddie Robinson    0.000664
517       0.0       Eddy Curry   -0.000428
...       ...              ...         ...
15645     0.0    Steve Francis    0.028624
15646     0.0   Terence Morris   -0.015437
15647     0.0     Tierre Brown   -0.003299
15648     0.0    Walt Williams   -0.013648
15649     0.0     Óscar Torres   -0.007015

[500 rows x 3 columns],
        Share           Player  prediction
523       0.0      A.J. Guyton    0.007825
524       0.0     Adonal Foyle    0.003776
525       0.0   Antawn Jamison    0.037919
526       0.0         Bob Sura    0.002127
527       0.0      Chris Mills   -0.003989
```

In [134]:
```
def add_ranks(combination):
    combination = combination.sort_values("Share",ascending=False)
    combination['Rk'] = list(range(1,combination.shape[0]+1))
    combination = combination.sort_values('prediction',ascending=False)
    combination['Predict_Rk'] = list(range(1,combination.shape[0]+1))
    combination['Diff'] = combination['Rk'] - combination['Predict_Rk']
    return combination
```

In [150]:
```
add_ranks(all_prediction[1]).query('Rk < 6').sort_values('Diff',ascending=False)
```

Out[150]:

	Share	Player	prediction	Rk	Predict_Rk	Diff
2642	0.857	Karl Malone	0.176051	1	2	-1
12663	0.832	Michael Jordan	0.154297	2	3	-1
16240	0.327	Grant Hill	0.125242	3	4	-1
5989	0.207	Tim Hardaway	0.050239	4	26	-22
10158	0.117	Glen Rice	0.019718	5	107	-102

In [193]:
```
def backtest(stats,model,year,predictors):
    aps = []
    all_prediction = []
    for year in years[5:]:
        train = df[df['Year'] < year]
        test = df[df['Year'] == year]
        model.fit(train[predictors],train['Share'])
        predictions = reg.predict(test[predictors])
        predictions = pd.DataFrame(predictions,columns=['prediction'],index = test.index)
        combination = pd.concat([test[['Share','Player']],predictions],axis=1)
        combination = add_ranks(combination)
        all_prediction.append(combination)
        aps.append(find_ap(combination))
    return sum(aps)/len(aps),aps,pd.concat(all_prediction)
```

In [234]:
```
mean_ap,aps,all_prediction = backtest(df,reg,years[5:],predictors)
```

In [191]:
```
mean_ap
```

Out[191]:
```
0.6944979748368247
```

- Diagnosing model performance

In [154]:
```
all_prediction[all_prediction['Rk']<=5].sort_values('Diff').head(10)
```

Out[154]:

	Share	Player	prediction	Rk	Predict_Rk	Diff
10158	0.117	Glen Rice	0.019718	5	107	-102
2055	0.712	Jason Kidd	0.023461	2	82	-80
6672	0.839	Steve Nash	0.036889	1	49	-48
5095	0.344	Chauncey Billups	0.044579	5	41	-36
14815	0.258	Joakim Noah	0.043823	4	39	-35
2450	0.138	Chris Paul	0.068735	5	36	-31
6687	0.739	Steve Nash	0.056451	1	30	-29
10379	0.228	Peja Stojaković	0.039963	4	29	-25
5989	0.207	Tim Hardaway	0.050239	4	26	-22
6701	0.785	Steve Nash	0.071394	2	21	-19

In [236]:
```
pd.concat([pd.Series(reg.coef_),pd.Series(predictors)],axis=1).sort_values(0,ascending=False).head(2)
```

Out[236]:

	0	1
22	0.103193	BLK
34	0.051508	PTS_R

- adding more predictors

In [165]:
```
df.columns
```

Out[165]:
```
Index(['Rk', 'Player', 'Pos', 'Age', 'Team', 'G', 'GS', 'MP', 'FG', 'FGA',
       'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA',
       'FT%', 'ORB', 'DRB', 'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PTS',
       'Year', 'Pts Won', 'Pts Max', 'Share', 'W', 'L', 'W/L%', 'GB', 'PS/G',
       'PA/G', 'SRS'],
      dtype='object')
```

In [179]:
```
df_ratios = df[['PTS','AST','STL','BLK','3P','Year']].groupby('Year').apply(lambda x:x/x.mean())
```

In [186]:
```
df[['PTS_R','AST_R','STL_R','BLK_R','3P_R']] = df_ratios[['PTS','AST','STL','BLK','3P']]
```

In [188]:
```
df.head()
```

Out[188]:

	Rk	Player	Pos	Age	Team	G	GS	MP	FG	FGA	...	W/L%	GB	PS/G	PA/G	SRS	PTS_R	AST_R	STL_R	BLK
0	135.0	A.C. Green	PF	27.0	LAL	82.0	21.0	26.4	3.1	6.6	...	0.707	5.0	106.3	99.6	6.73	1.019207	0.451512	0.984209	0.702
1	301.0	Byron Scott	SG	29.0	LAL	82.0	82.0	32.1	6.1	12.8	...	0.707	5.0	106.3	99.6	6.73	1.624011	1.103695	1.687215	0.702
2	51.0	Elden Campbell	PF	22.0	LAL	52.0	0.0	7.3	1.1	2.4	...	0.707	5.0	106.3	99.6	6.73	0.313602	0.100336	0.281203	1.640
3	330.0	Irving Thomas	PF	25.0	LAL	26.0	0.0	4.2	0.7	1.9	...	0.707	5.0	106.3	99.6	6.73	0.201601	0.200672	0.281203	0.000
4	384.0	James Worthy	SF	29.0	LAL	78.0	74.0	38.6	9.2	18.7	...	0.707	5.0	106.3	99.6	6.73	2.396817	1.755878	1.827817	0.937

5 rows × 46 columns

In [195]:
```
predictors += ['PTS_R','AST_R','STL_R','BLK_R','3P_R']
```

In [208]:
```
del predictors[-6]
```

In [238]:
```
predictors
```

Out[238]:
```
['Age',
 'G',
 'GS',
 'MP',
 'FG',
 'FGA',
 'FG%',
 '3P',
 '3PA',
 '3P%',
 '2P',
 '2PA',
 '2P%',
 'eFG%',
 'FT',
 'FTA',
 'FT%',
 'ORB',
 'DRB',
 'TRB',
 'AST',
 'STL',
 'BLK',
 'TOV',
 'PF',
 'PTS',
 'Year',
 'W',
 'L',
 'W/L%',
 'GB',
 'PS/G',
 'PA/G',
 'SRS',
 'PTS_R',
 'AST_R',
 'STL_R',
 'BLK_R',
 '3P_R']
```

In [210]:
```
mean_ap,aps,all_prediction = backtest(df,reg,years[5:],predictors)
```

In [211]:
```
mean_ap
```

Out[211]:
```
0.6933074406516423
```

In [218]:
```
df['NPos'] = df['Pos'].astype('category').cat.codes
```

In [220]:
```
df['NTm'] = df['Team'].astype('category').cat.codes
```

- using random forest

In [230]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=50,random_state=1,min_samples_split=5)
mean_ap1,aps,all_prediction = backtest(df,rf,years[28:],predictors)
```

In [231]:
```
mean_ap1
```

Out[231]:
```
0.7392204084879423
```

In [228]:
```
mean_ap,aps,all_prediction = backtest(df,reg,years[28:],predictors)
```

In [229]:
```
mean_ap
```

Out[229]:
```
0.6933074406516423
```

In [ ]:
```

```