In [1]:
```
pip install ISLP
```

```
Requirement already satisfied: ISLP in /usr/local/lib/python3.10/dist-packages (0.3.22)
Requirement already satisfied: numpy<1.25,>=1.7.1 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.24.4)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.11.4)
Requirement already satisfied: pandas<=1.9,>=0.20 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.5.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from ISLP) (4.9.4)
Requirement already satisfied: scikit-learn>=1.2 in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.4.2)
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.14.2)
Requirement already satisfied: lifelines in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.28.0)
Requirement already satisfied: pygam in /usr/local/lib/python3.10/dist-packages (from ISLP) (0.9.0)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.3.0+cu121)
Requirement already satisfied: pytorch-lightning in /usr/local/lib/python3.10/dist-packages (from ISLP) (2.2.5)
Requirement already satisfied: torchmetrics in /usr/local/lib/python3.10/dist-packages (from ISLP) (1.4.0.post0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas<=1.9,>=0.20->ISLP) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<=1.9,>=0.20->ISLP) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.2->ISLP) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13->ISLP) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13->ISLP) (24.0)
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (3.7.1)
Requirement already satisfied: autograd>=1.5 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (1.6.2)
Requirement already satisfied: autograd-gamma>=0.3 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (0.5.0)
Requirement already satisfied: formulaic>=0.2.2 in /usr/local/lib/python3.10/dist-packages (from lifelines->ISLP) (1.0.1)
Requirement already satisfied: progressbar2<5.0.0,>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pygam->ISLP) (4.2.0)
Requirement already satisfied: tqdm>=4.57.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (4.66.4)
Requirement already satisfied: PyYAML>=5.4 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (6.0.1)
Requirement already satisfied: fsspec[http]>=2022.5.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (2023.6.0)
Requirement already satisfied: typing-extensions>=4.4.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (4.11.0)
Requirement already satisfied: lightning-utilities>=0.8.0 in /usr/local/lib/python3.10/dist-packages (from pytorch-lightning->ISLP) (0.11.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.14.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (3.1.4)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.20.5 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (2.20.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (12.1.105)
Requirement already satisfied: triton==2.3.0 in /usr/local/lib/python3.10/dist-packages (from torch->ISLP) (2.3.0)
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-cu12==11.4.5.107->torch->ISLP) (12.5.40)
Requirement already satisfied: future>=0.15.2 in /usr/local/lib/python3.10/dist-packages (from autograd>=1.5->lifelines->ISLP) (0.18.3)
Requirement already satisfied: interface-meta>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.2.2->lifelines->ISLP) (1.3.0)
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.2.2->lifelines->ISLP) (1.14.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (2.31.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /usr/local/lib/python3.10/dist-packages (from fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.9.5)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from lightning-utilities>=0.8.0->pytorch-lightning->ISLP) (67.7.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0->lifelines->ISLP) (3.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels>=0.13->ISLP) (1.16.0)
Requirement already satisfied: python-utils>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from progressbar2<5.0.0,>=4.2.0->pygam->ISLP) (3.8.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch->ISLP) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch->ISLP) (1.3.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (4.0.
3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->fsspec[http]>=2022.5.0->pytorch-lightning->ISLP) (2024.2.2)
```

In [2]:
```
import numpy as np
import pandas as pd
from matplotlib.pyplot import subplots
from statsmodels.api import OLS
import sklearn.model_selection as skm
import sklearn.linear_model as skl
from sklearn.preprocessing import StandardScaler
from ISLP import load_data
from ISLP.models import ModelSpec as MS
from functools import partial
```

In [3]:
```
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
from ISLP.models import \
     (Stepwise,
      sklearn_selected,
      sklearn_selection_path)
```

In [4]:
```
from ISLP import load_data
from ISLP.models import (ModelSpec as MS,
                         summarize,
                         poly)
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
```

In [5]:
```
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving nba_salary2022.csv to nba_salary2022.csv
```

In [6]:
```
stats = pd.read_csv('nba_salary2022.csv')
stats
```

Out[6]:

	Unnamed: 0	Player Name	Salary	Position	Age	Team	GP	GS	MP	FG	...	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM	DBPM	BPM	VORP
0	0	Stephen Curry	48070014	PG	34	GSW	56	56	34.7	10.0	...	12.5	31.0	5.8	2.0	7.8	0.192	7.5	0.1	7.5	4.7
1	1	John Wall	47345760	PG	32	LAC	34	3	22.2	4.1	...	17.1	27.0	-0.4	0.7	0.3	0.020	-0.8	-0.4	-1.2	0.1
2	2	Russell Westbrook	47080179	PG	34	LAL/LAC	73	24	29.1	5.9	...	18.4	27.7	-0.6	2.6	1.9	0.044	0.3	-0.1	0.2	1.2
3	3	LeBron James	44474988	PF	38	LAL	55	54	35.5	11.1	...	11.6	33.3	3.2	2.4	5.6	0.138	5.5	0.6	6.1	4.0
4	4	Kevin Durant	44119845	PF	34	BRK/PHO	47	47	35.6	10.3	...	13.4	30.7	4.7	2.1	6.8	0.194	6.0	1.2	7.1	3.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
462	462	Justin Minaya	35096	SF	23	POR	4	0	22.3	1.8	...	14.6	13.4	-0.2	0.1	-0.1	-0.067	-7.2	-1.9	-9.0	-0.2
463	463	Kobi Simmons	32795	SG	25	CHO	5	0	5.6	0.2	...	12.7	11.8	0.0	0.0	0.0	0.019	-1.0	0.1	-0.9	0.0
464	464	Gabe York	32171	SG	29	IND	3	0	18.7	2.7	...	0.0	16.4	0.1	0.0	0.1	0.091	-1.7	-1.8	-3.5	0.0
465	465	RaiQuan Gray	5849	PF	23	BRK	1	0	35.0	6.0	...	23.7	21.4	0.0	0.0	0.1	0.106	-0.6	-1.4	-2.0	0.0
466	466	Jacob Gilyard	5849	PG	24	MEM	1	0	41.0	1.0	...	40.0	5.1	0.0	0.1	0.1	0.079	-7.8	1.7	-6.1	0.0

467 rows × 52 columns

In [7]:

```
stats = stats.drop(columns=['Unnamed: 0', 'Player Name'], axis =1)
stats
```

Out[7]:

	Salary	Position	Age	Team	GP	GS	MP	FG	FGA	FG%	...	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM	DBPM	BPM	VORP
0	48070014	PG	34	GSW	56	56	34.7	10.0	20.2	0.493	...	12.5	31.0	5.8	2.0	7.8	0.192	7.5	0.1	7.5	4.7
1	47345760	PG	32	LAC	34	3	22.2	4.1	9.9	0.408	...	17.1	27.0	-0.4	0.7	0.3	0.020	-0.8	-0.4	-1.2	0.1
2	47080179	PG	34	LAL/LAC	73	24	29.1	5.9	13.6	0.436	...	18.4	27.7	-0.6	2.6	1.9	0.044	0.3	-0.1	0.2	1.2
3	44474988	PF	38	LAL	55	54	35.5	11.1	22.2	0.500	...	11.6	33.3	3.2	2.4	5.6	0.138	5.5	0.6	6.1	4.0
4	44119845	PF	34	BRK/PHO	47	47	35.6	10.3	18.3	0.560	...	13.4	30.7	4.7	2.1	6.8	0.194	6.0	1.2	7.1	3.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
462	35096	SF	23	POR	4	0	22.3	1.8	5.8	0.304	...	14.6	13.4	-0.2	0.1	-0.1	-0.067	-7.2	-1.9	-9.0	-0.2
463	32795	SG	25	CHO	5	0	5.6	0.2	1.2	0.167	...	12.7	11.8	0.0	0.0	0.0	0.019	-1.0	0.1	-0.9	0.0
464	32171	SG	29	IND	3	0	18.7	2.7	7.0	0.381	...	0.0	16.4	0.1	0.0	0.1	0.091	-1.7	-1.8	-3.5	0.0
465	5849	PF	23	BRK	1	0	35.0	6.0	12.0	0.500	...	23.7	21.4	0.0	0.0	0.1	0.106	-0.6	-1.4	-2.0	0.0
466	5849	PG	24	MEM	1	0	41.0	1.0	3.0	0.333	...	40.0	5.1	0.0	0.1	0.1	0.079	-7.8	1.7	-6.1	0.0

467 rows × 50 columns

In [8]:

```
idx_20 = stats['GP'] > 20
newStats = stats[idx_20]
newStats
```

Out[8]:

	Salary	Position	Age	Team	GP	GS	MP	FG	FGA	FG%	...	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM	DBPM	BPM	VORP
0	48070014	PG	34	GSW	56	56	34.7	10.0	20.2	0.493	...	12.5	31.0	5.8	2.0	7.8	0.192	7.5	0.1	7.5	4.7
1	47345760	PG	32	LAC	34	3	22.2	4.1	9.9	0.408	...	17.1	27.0	-0.4	0.7	0.3	0.020	-0.8	-0.4	-1.2	0.1
2	47080179	PG	34	LAL/LAC	73	24	29.1	5.9	13.6	0.436	...	18.4	27.7	-0.6	2.6	1.9	0.044	0.3	-0.1	0.2	1.2
3	44474988	PF	38	LAL	55	54	35.5	11.1	22.2	0.500	...	11.6	33.3	3.2	2.4	5.6	0.138	5.5	0.6	6.1	4.0
4	44119845	PF	34	BRK/PHO	47	47	35.6	10.3	18.3	0.560	...	13.4	30.7	4.7	2.1	6.8	0.194	6.0	1.2	7.1	3.9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418	508891	C	24	MIN	28	0	8.7	2.3	4.1	0.543	...	9.2	26.9	0.9	0.2	1.1	0.211	2.2	-2.6	-0.4	0.1
419	508891	SG	25	ORL	34	0	13.4	1.4	3.1	0.439	...	14.4	13.7	0.2	0.4	0.6	0.065	-3.5	-0.1	-3.6	-0.2
422	508891	PF	19	SAS	28	0	14.6	1.6	3.1	0.535	...	13.4	11.3	0.5	0.2	0.7	0.082	-3.7	-0.8	-4.6	-0.3
424	508891	PG	25	TOR	25	0	10.4	1.0	2.3	0.439	...	7.6	10.9	0.2	0.2	0.4	0.076	-2.8	0.9	-1.9	0.0
430	386055	C	22	MIA	31	1	13.7	1.5	2.9	0.528	...	13.5	12.4	0.5	0.6	1.1	0.129	-2.7	0.7	-2.1	0.0

374 rows × 50 columns

In [9]:

```
stats1 = newStats.iloc[:,[0,1,2,4,5,6,7,9,10,12,13,15,23,28,29]]
stats1
```

Out[9]:

	Salary	Position	Age	GP	GS	MP	FG	FG%	3P	3P%	2P	2P%	AST	PTS	Total Minutes
0	48070014	PG	34	56	56	34.7	10.0	0.493	4.9	0.427	5.1	0.579	6.3	29.4	1941
1	47345760	PG	32	34	3	22.2	4.1	0.408	1.0	0.303	3.1	0.459	5.2	11.4	755
2	47080179	PG	34	73	24	29.1	5.9	0.436	1.2	0.311	4.7	0.487	7.5	15.9	2126
3	44474988	PF	38	55	54	35.5	11.1	0.500	2.2	0.321	8.9	0.580	6.8	28.9	1954
4	44119845	PF	34	47	47	35.6	10.3	0.560	2.0	0.404	8.3	0.617	5.0	29.1	1672
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418	508891	C	24	28	0	8.7	2.3	0.543	0.5	0.359	1.8	0.636	0.6	6.5	243
419	508891	SG	25	34	0	13.4	1.4	0.439	0.5	0.372	0.9	0.484	0.5	4.1	457
422	508891	PF	19	28	0	14.6	1.6	0.535	0.0	0.000	1.6	0.561	0.9	3.9	408
424	508891	PG	25	25	0	10.4	1.0	0.439	0.2	0.313	0.8	0.488	1.2	2.4	259
430	386055	C	22	31	1	13.7	1.5	0.528	0.0	0.000	1.5	0.566	0.8	3.7	425

374 rows × 15 columns

In [10]:

```
y = stats1['Salary']
logy = np.log(y)
```

In [11]:

```
dummies = pd.get_dummies(stats1.Position)
stats1['logSalary']=logy
nba = pd.concat([stats1, dummies], axis='columns')
nba
```

```
<ipython-input-11-597aebfac545>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  stats1['logSalary']=logy
```

Out[11]:

	Salary	Position	Age	GP	GS	MP	FG	FG%	3P	3P%	...	2P%	AST	PTS	Total Minutes	logSalary	C	PF	PG	SF	SG
0	48070014	PG	34	56	56	34.7	10.0	0.493	4.9	0.427	...	0.579	6.3	29.4	1941	17.688169	0	0	1	0	0
1	47345760	PG	32	34	3	22.2	4.1	0.408	1.0	0.303	...	0.459	5.2	11.4	755	17.672988	0	0	1	0	0
2	47080179	PG	34	73	24	29.1	5.9	0.436	1.2	0.311	...	0.487	7.5	15.9	2126	17.667363	0	0	1	0	0
3	44474988	PF	38	55	54	35.5	11.1	0.500	2.2	0.321	...	0.580	6.8	28.9	1954	17.610438	0	1	0	0	0
4	44119845	PF	34	47	47	35.6	10.3	0.560	2.0	0.404	...	0.617	5.0	29.1	1672	17.602420	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418	508891	C	24	28	0	8.7	2.3	0.543	0.5	0.359	...	0.636	0.6	6.5	243	13.139989	1	0	0	0	0
419	508891	SG	25	34	0	13.4	1.4	0.439	0.5	0.372	...	0.484	0.5	4.1	457	13.139989	0	0	0	0	1
422	508891	PF	19	28	0	14.6	1.6	0.535	0.0	0.000	...	0.561	0.9	3.9	408	13.139989	0	1	0	0	0
424	508891	PG	25	25	0	10.4	1.0	0.439	0.2	0.313	...	0.488	1.2	2.4	259	13.139989	0	0	1	0	0
430	386055	C	22	31	1	13.7	1.5	0.528	0.0	0.000	...	0.566	0.8	3.7	425	12.863735	1	0	0	0	0

374 rows × 21 columns

In [12]:

```
nba = nba.dropna()
nba
```

Out[12]:

	Salary	Position	Age	GP	GS	MP	FG	FG%	3P	3P%	...	2P%	AST	PTS	Total Minutes	logSalary	C	PF	PG	SF	SG
0	48070014	PG	34	56	56	34.7	10.0	0.493	4.9	0.427	...	0.579	6.3	29.4	1941	17.688169	0	0	1	0	0
1	47345760	PG	32	34	3	22.2	4.1	0.408	1.0	0.303	...	0.459	5.2	11.4	755	17.672988	0	0	1	0	0
2	47080179	PG	34	73	24	29.1	5.9	0.436	1.2	0.311	...	0.487	7.5	15.9	2126	17.667363	0	0	1	0	0
3	44474988	PF	38	55	54	35.5	11.1	0.500	2.2	0.321	...	0.580	6.8	28.9	1954	17.610438	0	1	0	0	0
4	44119845	PF	34	47	47	35.6	10.3	0.560	2.0	0.404	...	0.617	5.0	29.1	1672	17.602420	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
418	508891	C	24	28	0	8.7	2.3	0.543	0.5	0.359	...	0.636	0.6	6.5	243	13.139989	1	0	0	0	0
419	508891	SG	25	34	0	13.4	1.4	0.439	0.5	0.372	...	0.484	0.5	4.1	457	13.139989	0	0	0	0	1
422	508891	PF	19	28	0	14.6	1.6	0.535	0.0	0.000	...	0.561	0.9	3.9	408	13.139989	0	1	0	0	0
424	508891	PG	25	25	0	10.4	1.0	0.439	0.2	0.313	...	0.488	1.2	2.4	259	13.139989	0	0	1	0	0
430	386055	C	22	31	1	13.7	1.5	0.528	0.0	0.000	...	0.566	0.8	3.7	425	12.863735	1	0	0	0	0

367 rows × 21 columns

In [13]:

```
nba.shape
```

Out[13]:

```
(367, 21)
```

In [14]:

```
nba.columns
```

Out[14]:

```
Index(['Salary', 'Position', 'Age', 'GP', 'GS', 'MP', 'FG', 'FG%', '3P', '3P%',
       '2P', '2P%', 'AST', 'PTS', 'Total Minutes', 'logSalary', 'C', 'PF',
       'PG', 'SF', 'SG'],
      dtype='object')
```

In [15]:

```
nba.describe()
```

Out[15]:

	Salary	Age	GP	GS	MP	FG	FG%	3P	3P%	2P	2P%	AST	PTS	Total Minutes	logSalary	C	PF
count	3.670000e+02	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000	367.000000
mean	1.010955e+07	26.016349	58.226158	28.128065	22.246322	3.834605	0.472049	1.142507	0.331226	2.693733	0.540414	2.389373	10.488283	1369.632153	15.556817	0.190736	0.190736
std	1.123308e+07	4.302719	16.200775	27.727646	8.535399	2.445175	0.075910	0.886568	0.112337	2.033957	0.076075	1.968820	6.929536	726.872136	1.090890	0.393417	0.393417
min	3.860550e+05	19.000000	22.000000	0.000000	4.700000	0.300000	0.259000	0.000000	0.000000	0.100000	0.286000	0.200000	0.900000	107.000000	12.863735	0.000000	0.000000
25%	2.271820e+06	23.000000	46.000000	3.000000	15.100000	2.000000	0.425500	0.500000	0.307000	1.200000	0.494500	1.000000	5.300000	767.500000	14.636089	0.000000	0.000000
50%	5.155500e+06	25.000000	62.000000	16.000000	21.700000	3.200000	0.457000	1.000000	0.350000	2.000000	0.539000	1.600000	8.700000	1258.000000	15.455575	0.000000	0.000000
75%	1.343741e+07	29.000000	72.000000	58.000000	30.000000	5.100000	0.508000	1.700000	0.385500	3.750000	0.587000	3.400000	13.750000	1986.000000	16.413527	0.000000	0.000000
max	4.807001e+07	38.000000	83.000000	83.000000	37.400000	11.200000	0.776000	4.900000	1.000000	10.500000	0.783000	10.700000	33.100000	2963.000000	17.688169	1.000000	1.000000

(a) Split the data set into a training set and a test set.

In [16]:

```
nba_train, nba_valid = train_test_split(nba,
                                        test_size=0.5,
                                        random_state=0)
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

In [17]:

```
allvars = nba.columns.drop(['Salary', 'Position', 'logSalary'])
design = MS(allvars)
X_train = design.fit_transform(nba_train)
y_train = nba_train['logSalary']
model = sm.OLS(y_train, X_train,family=sm.families.Binomial())
results = model.fit()
summarize(results)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:130: ValueWarning: unknown kwargs ['family']
  warnings.warn(msg, ValueWarning)
```

Out[17]:

	coef	std err	t	P>|t|
intercept	9.6726	0.567	17.060	0.000
Age	0.0928	0.013	7.226	0.000
GP	0.0043	0.011	0.395	0.693
GS	0.0037	0.005	0.756	0.451
MP	0.0527	0.027	1.977	0.050
FG	0.3458	1.173	0.295	0.769
FG%	0.9588	1.568	0.612	0.542
3P	-0.1951	1.158	-0.168	0.866
3P%	-0.2986	0.459	-0.651	0.516
2P	-0.1431	1.159	-0.124	0.902
2P%	-1.1184	1.282	-0.872	0.384
AST	0.0032	0.051	0.062	0.951
PTS	-0.0101	0.067	-0.150	0.881
Total Minutes	-0.0003	0.001	-0.556	0.579
C	1.8668	0.214	8.743	0.000
PF	1.8796	0.160	11.722	0.000
PG	2.0252	0.172	11.772	0.000
SF	1.9718	0.154	12.837	0.000
SG	1.9292	0.134	14.431	0.000

Age, MP, C,PF,SG, PG, SF

In [18]:
```
X_valid = design.transform(nba_valid)
y_valid = nba_valid['logSalary']
valid_pred = results.predict(X_valid)
q = np.mean((y_valid - valid_pred)**2)
q
```
Out[18]:
```
0.3948359074395269
```

Ridge Regression

In [19]:
```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_squared_error

ridge_cv = RidgeCV(alphas=[0.1, 1.0, 10.0], cv=5)
ridge_cv.fit(X_train, y_train)
y_ridge = ridge_cv.predict(X_valid)
ridge_error = mean_squared_error(y_valid, y_ridge)
w = ridge_error
w
```
Out[19]:
```
0.3911805145024713
```

The test error is 0.3911

(d) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefcient estimates.

In [20]:
```
from sklearn.linear_model import LassoCV

lasso_cv = LassoCV(alphas=[0.1, 1.0, 10.0], cv=5)
lasso_cv.fit(X_train, y_train)
y_lasso = lasso_cv.predict(X_valid)
lasso_error = mean_squared_error(y_valid, y_lasso)
lasso_error
```
Out[20]:
```
0.3889050920577201
```

In [21]:
```
nzc = np.sum(lasso_cv.coef_ != 0)
nzc
```
Out[21]:
```
6
```

In [22]:
```
lasso_cv.coef_
```
Out[22]:
```
array([ 0.00000000e+00,  8.26936213e-02, -1.16395153e-03,  3.71637312e-03,
        4.84960325e-02,  0.00000000e+00,  0.00000000e+00, -0.00000000e+00,
       -0.00000000e+00,  0.00000000e+00, -0.00000000e+00,  0.00000000e+00,
        5.01219888e-02, -8.56377761e-05,  0.00000000e+00, -0.00000000e+00,
        0.00000000e+00,  0.00000000e+00, -0.00000000e+00])
```

The test error for the lasso model is 0.3889.

The number of non-zero coefficients is 6.

(e) Fit a PCR model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

In [23]:
```
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('pca', PCA()),
    ('linear_regression', LinearRegression())
])

param_grid = {
    'pca__n_components': [1, 2, 3, 4, 5]
}
grid = GridSearchCV(pipe,
                    param_grid,
                    cv=5,
                    scoring='neg_mean_squared_error')
grid.fit(X_train, y_train)

bestPCA = grid.best_estimator_
bestPCA.fit(X_train, y_train)

y_pcr = bestPCA.predict(X_valid)
pcr_error = mean_squared_error(y_valid, y_pcr)
pcr_error
```
Out[23]:
```
0.5479620198331824
```

In [24]:
```
m = bestPCA.named_steps['pca'].n_components_
m
```
Out[24]:
```
5
```

The test error for the PCR model is 0.5479.

M = 5

(f) Fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation

In [25]:
```
from sklearn.cross_decomposition import PLSRegression

pls = PLSRegression()
grid1 = {'n_components': [1, 2, 3, 4, 5]}

grid2 = GridSearchCV(pls, grid1, cv=5, scoring='neg_mean_squared_error')
grid2.fit(X_train, y_train)

bestPLS = grid2.best_estimator_
bestPLS.fit(X_train, y_train)

y_pls = bestPLS.predict(X_valid)
pls_error = mean_squared_error(y_valid, y_pls)
pls_error
```

Out[25]:
```
0.4269552224184196
```

In [26]:
```
m2 = grid2.best_params_['n_components']
m2
```

Out[26]:
```
2
```

The test error for the PLS model is 47347967999570.85 .

M = 2.

Test errors:

Least Squares - 44,277,780,198,311.74

Ridge - 43,677,551,763,662.33

Lasso - 44,269,681,361,453.77

PCR - 58,543,756,710,948.35

PLS - 47,347,967,999,570.85

Ridge has the lowest test error. PCR had the highest test error.

Refitting the model based on forward selection.

In [27]:
```
from ISLP.models import \
     (Stepwise,
      sklearn_selected,
      sklearn_selection_path)
```

In [28]:
```
def nCp(sigma2, estimator, X, Y):
    "Negative Cp statistic"
    n, p = X.shape
    Yhat = estimator.predict(X)
    RSS = np.sum((Y - Yhat)**2)
    return -(RSS + 2 * p * sigma2) / n
```

In [29]:
```
allvars = nba.columns.drop(['Salary', 'Position', 'logSalary'])
design = MS(allvars).fit(nba)
Y = np.array(nba['logSalary'])
X = design.transform(nba)
sigma2 = OLS(Y,X).fit().scale
```

In [30]:
```
neg_Cp = partial(nCp, sigma2)
```

In [31]:
```
strategy = Stepwise.first_peak(design,
                               direction='forward',
                               max_terms=len(design.terms))
```

In [32]:
```
nba_Cp = sklearn_selected(OLS,
                          strategy,
                          scoring=neg_Cp)
nba_Cp.fit(nba, Y)
nba_Cp.selected_state_
```

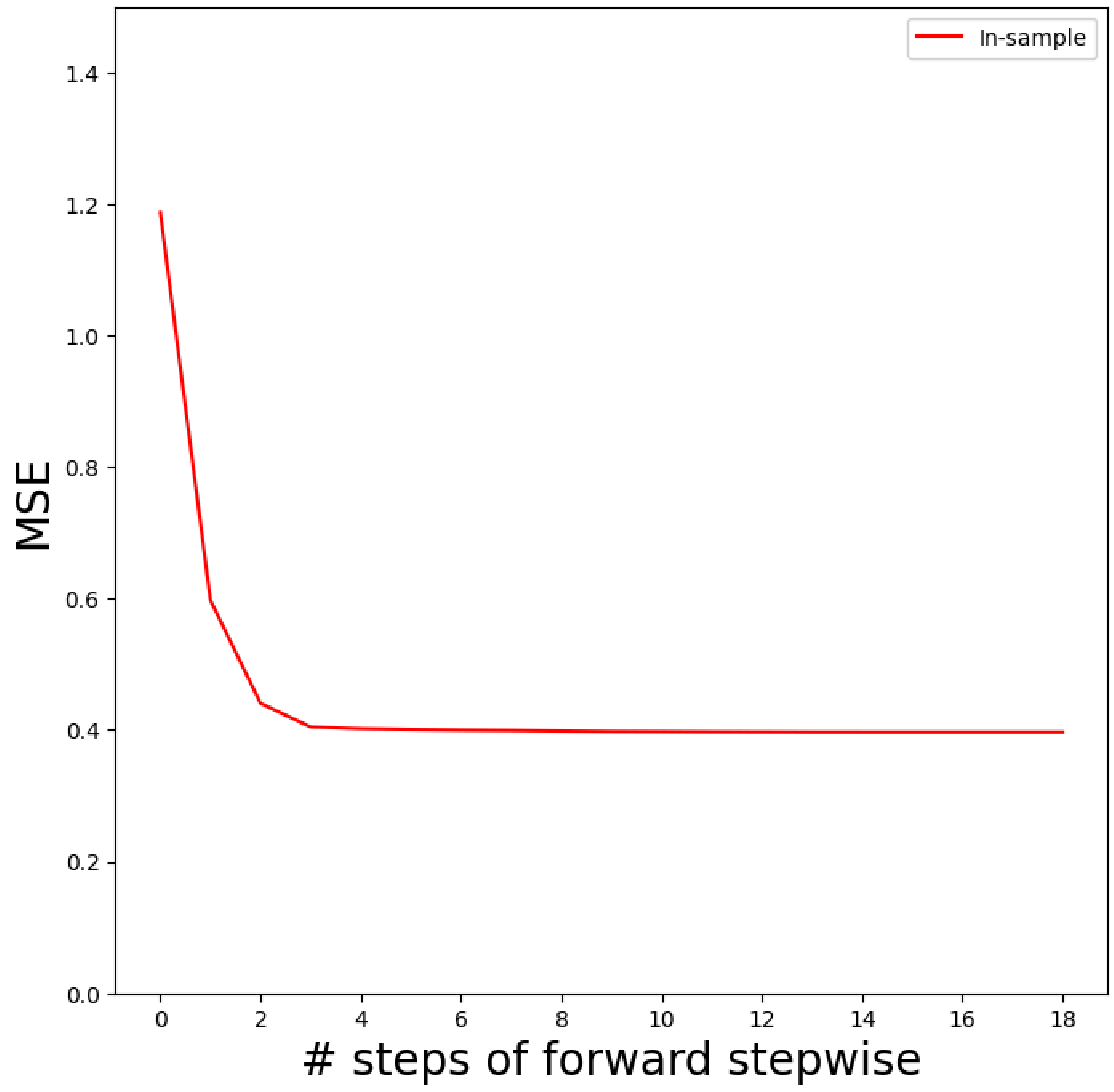
Out[32]:
```
('3P%', 'Age', 'FG', 'MP')
```

In [33]:
```
strategy = Stepwise.fixed_steps(design,
                                len(design.terms),
                                direction='forward')
full_path = sklearn_selection_path(OLS, strategy)
```

In [34]:
```
full_path.fit(nba, Y)
Yhat_in = full_path.predict(nba)
Yhat_in.shape
```

Out[34]:
```
(367, 19)
```

In [35]:
```
mse_fig, ax = subplots(figsize=(8,8))
insample_mse = ((Yhat_in - Y[:,None])**2).mean(0)
n_steps = insample_mse.shape[0]
ax.plot(np.arange(n_steps),
        insample_mse,
        'r', # color red
        label='In-sample')
ax.set_ylabel('MSE',
              fontsize=20)
ax.set_xlabel('# steps of forward stepwise',
              fontsize=20)
ax.set_xticks(np.arange(n_steps)[::2])
ax.legend()
ax.set_ylim([0,1.5]);
```



Linear Regression

In [36]:
```
nba_train, nba_valid = train_test_split(nba,
                                         test_size=0.5,
                                         random_state=0)
```

In [37]:
```
design2 = MS(['Age','FG','MP','3P%'])
X_train2 = design2.fit_transform(nba_train)
y_train2 = nba_train['logSalary']
model2 = sm.OLS(y_train2, X_train2,family=sm.families.Binomial())
results2 = model2.fit()
summarize(results2)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/base/model.py:130: ValueWarning: unknown kwargs ['family']
  warnings.warn(msg, ValueWarning)
```

Out[37]:

	coef	std err	t	P>|t|
intercept	11.5910	0.318	36.485	0.000
Age	0.0927	0.012	7.908	0.000
FG	0.1684	0.039	4.274	0.000
MP	0.0456	0.012	3.948	0.000
3P%	-0.3968	0.407	-0.976	0.331

In [38]:
```
X_valid = design2.transform(nba_valid)
y_valid = nba_valid['logSalary']
valid_pred = results2.predict(X_valid)
np.mean((y_valid - valid_pred)**2)
```

Out[38]: 0.38463875496623307

Ridge Regression

In [39]:
```
from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_squared_error

ridge_cv = RidgeCV(alphas=[0.1, 1.0, 10.0], cv=5)
ridge_cv.fit(X_train2, y_train2)
y_ridge = ridge_cv.predict(X_valid)
ridge_error = mean_squared_error(y_valid, y_ridge)
ridge_error
```

Out[39]: 0.3868427501095814

Lasso Regression

In [40]:
```
from sklearn.linear_model import LassoCV

lasso_cv = LassoCV(alphas=[0.1, 1.0, 10.0], cv=5)
lasso_cv.fit(X_train2, y_train2)
y_lasso = lasso_cv.predict(X_valid)
lasso_error = mean_squared_error(y_valid, y_lasso)
lasso_error
```

Out[40]: 0.39080239062267963

In [41]:
```
nzc = np.sum(lasso_cv.coef_ != 0)
nzc
```

Out[41]: 3

In [42]:
```
lasso_cv.coef_
```

Out[42]:
```
array([ 0.        ,  0.08475921,  0.11682309,  0.05717262, -0.        ])
```

PCR Model

In [43]:
```
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('pca', PCA()),
    ('linear_regression', LinearRegression())
])

param_grid = {
    'pca__n_components': [1, 2, 3, 4, 5]
}
grid = GridSearchCV(pipe,
                    param_grid,
```

```
                    cv=5,
                    scoring='neg_mean_squared_error')
grid.fit(X_train2, y_train2)

bestPCA = grid.best_estimator_
bestPCA.fit(X_train2, y_train2)

y_pcr = bestPCA.predict(X_valid)
pcr_error = mean_squared_error(y_valid, y_pcr)
pcr_error
```

Out[43]:
```
0.3845788408198396
```

In [44]:
```
m = bestPCA.named_steps['pca'].n_components_
m
```

Out[44]:
```
3
```

PLS Model

In [45]:
```
from sklearn.cross_decomposition import PLSRegression

pls = PLSRegression()
grid1 = {'n_components': [1, 2, 3, 4, 5]}

grid2 = GridSearchCV(pls, grid1, cv=5, scoring='neg_mean_squared_error')
grid2.fit(X_train2, y_train2)

bestPLS = grid2.best_estimator_
bestPLS.fit(X_train2, y_train2)

y_pls = bestPLS.predict(X_valid)
pls_error = mean_squared_error(y_valid, y_pls)
pls_error
```

Out[45]:
```
0.3847887018256699
```

In [46]:
```
m2 = grid2.best_params_['n_components']
m2
```

Out[46]:
```
3
```

Test errors:

Least Squares - 42,209,786,173,676.555

Ridge - 42,465,327,591,902.73

Lasso - 42,209,785,969,552.3

PCR - 42,209,786,173,676.55

PLS - 42,072,278,406,218.2

PLS has the lowest test error. Ridge had the highest test error.

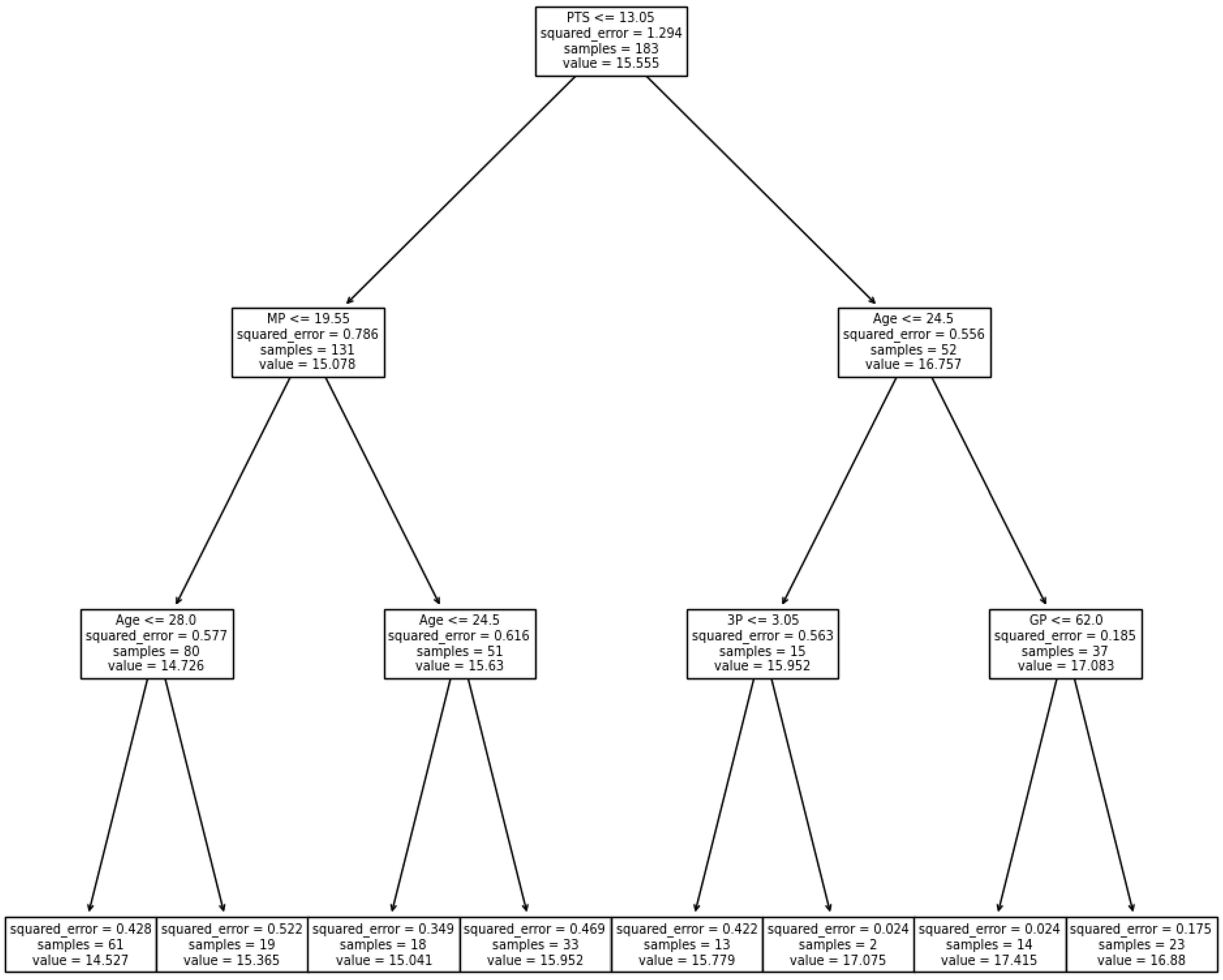
Decision Trees

In [47]:
```
from sklearn.tree import (DecisionTreeClassifier as DTC,
                          DecisionTreeRegressor as DTR,
                          plot_tree,
                          export_text)
from sklearn.metrics import (accuracy_score,
                             log_loss)
from sklearn.ensemble import \
     (RandomForestRegressor as RF,
      GradientBoostingRegressor as GBR)
from ISLP.bart import BART
```

In [48]:
```
model = MS(nba.columns.drop(['Salary', 'Position', 'logSalary']), intercept=False)
D = model.fit_transform(nba)
feature_names = list(D.columns)
X = np.asarray(D)
```

In [49]:
```
(X_train,
 X_test,
 y_train,
 y_test) = skm.train_test_split(X,
                                nba['logSalary'],
                                test_size=0.5,
                                random_state=0)
```

In [50]:
```
reg = DTR(max_depth=3)
reg.fit(X_train, y_train)
ax = subplots(figsize=(12,12))[1]
plot_tree(reg,
          feature_names=feature_names,
          ax=ax);
```



In [51]:
```
y_hat = reg.predict(X_test)
b = np.mean((y_test - y_hat)**2)
b
```

Out[51]:
```
0.5648409747890684
```

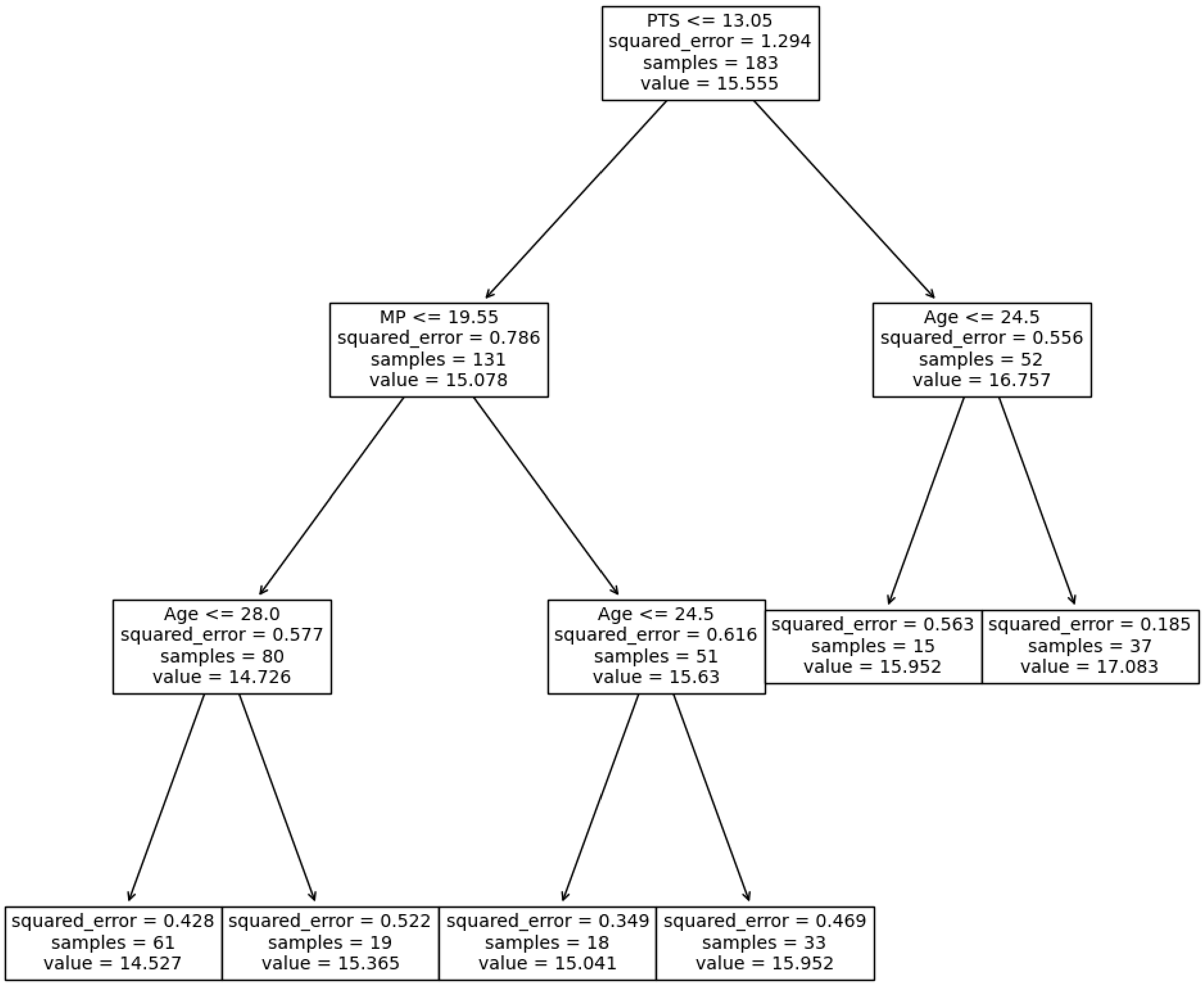
Pruning using Cross-Validation

In [52]:
```
ccp_path = reg.cost_complexity_pruning_path(X_train, y_train)
kfold = skm.KFold(5,
                  shuffle=True,
                  random_state=10)
grid = skm.GridSearchCV(reg,
                        {'ccp_alpha': ccp_path.ccp_alphas},
                        refit=True,
                        cv=kfold,
                        scoring='neg_mean_squared_error')
G = grid.fit(X_train, y_train)
```

In [53]:
```
best_ = grid.best_estimator_
a = np.mean((y_test - best_.predict(X_test))**2)
a
```

Out[53]:
```
0.5263873203509902
```

In [54]:
```
ax = subplots(figsize=(12,12))[1]
plot_tree(G.best_estimator_,
          feature_names=feature_names,
          ax=ax);
```


PTS <= 13.05
squared_error = 1.294
samples = 183
value = 15.555
MP <= 19.55
squared_error = 0.786
samples = 131
value = 15.078
Age <= 24.5
squared_error = 0.556
samples = 52
value = 16.757
Age <= 28.0
squared_error = 0.577
samples = 80
value = 14.726
Age <= 24.5
squared_error = 0.616
samples = 51
value = 15.63
squared_error = 0.563
samples = 15
value = 15.952
squared_error = 0.185
samples = 37
value = 17.083
squared_error = 0.428
samples = 61
value = 14.527
squared_error = 0.522
samples = 19
value = 15.365
squared_error = 0.349
samples = 18
value = 15.041
squared_error = 0.469
samples = 33
value = 15.952