Chapter 6 - Linear Model Selection and Regularization

September 14, 2023

[97]:
```
!pip install ISLP
```

```
Requirement already satisfied: ISLP in
/Users/barnana/anaconda3/lib/python3.10/site-packages (0.3.16)
Requirement already satisfied: pandas>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.5.3)
Requirement already satisfied: numpy>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.25.1)
Requirement already satisfied: pygam>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (0.9.0)
Requirement already satisfied: scipy>=0.9 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.11.1)
Requirement already satisfied: jupyter>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.0.0)
Requirement already satisfied: statsmodels>=0.13 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (0.13.5)
Requirement already satisfied: lifelines>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (0.27.7)
Requirement already satisfied: scikit-learn>=1.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.3.0)
Requirement already satisfied: joblib>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (1.1.1)
Requirement already satisfied: matplotlib>=3.3.3 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (3.7.0)
Requirement already satisfied: lxml>=0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from ISLP) (4.9.1)
Requirement already satisfied: ipykernel in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(6.19.2)
Requirement already satisfied: nbconvert in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(6.5.4)
Requirement already satisfied: qtconsole in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(5.4.0)
Requirement already satisfied: jupyter-console in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(6.6.3)
```

```
Requirement already satisfied: ipywidgets in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(8.0.7)
Requirement already satisfied: notebook in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter>=0.0->ISLP)
(6.5.2)
Requirement already satisfied: autograd>=1.5 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
lifelines>=0.0->ISLP) (1.6.2)
Requirement already satisfied: formulaic>=0.2.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
lifelines>=0.0->ISLP) (0.6.4)
Requirement already satisfied: autograd-gamma>=0.3 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
lifelines>=0.0->ISLP) (0.5.0)
Requirement already satisfied: python-dateutil>=2.7 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (2.8.2)
Requirement already satisfied: cycler>=0.10 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (9.4.0)
Requirement already satisfied: packaging>=20.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (22.0)
Requirement already satisfied: contourpy>=1.0.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
matplotlib>=3.3.3->ISLP) (4.25.0)
Requirement already satisfied: pytz>=2020.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pandas>=0.0->ISLP)
(2022.7)
Requirement already satisfied: progressbar2<5.0.0,>=4.2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pygam>=0.0->ISLP)
(4.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from scikit-
learn>=1.2->ISLP) (3.2.0)
```

```
Requirement already satisfied: patsy>=0.5.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
statsmodels>=0.13->ISLP) (0.5.3)
Requirement already satisfied: future>=0.15.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
autograd>=1.5->lifelines>=0.0->ISLP) (0.18.3)
Requirement already satisfied: wrapt>=1.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
formulaic>=0.2.2->lifelines>=0.0->ISLP) (1.14.1)
Requirement already satisfied: typing-extensions>=4.2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
formulaic>=0.2.2->lifelines>=0.0->ISLP) (4.4.0)
Requirement already satisfied: astor>=0.8 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
formulaic>=0.2.2->lifelines>=0.0->ISLP) (0.8.1)
Requirement already satisfied: interface-meta>=1.2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
formulaic>=0.2.2->lifelines>=0.0->ISLP) (1.3.0)
Requirement already satisfied: six in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
patsy>=0.5.2->statsmodels>=0.13->ISLP) (1.16.0)
Requirement already satisfied: python-utils>=3.0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
progressbar2<5.0.0,>=4.2.0->pygam>=0.0->ISLP) (3.7.0)
Requirement already satisfied: comm>=0.1.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (0.1.2)
Requirement already satisfied: ipython>=7.23.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (8.10.0)
Requirement already satisfied: pyzmq>=17 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (23.2.0)
Requirement already satisfied: psutil in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (5.9.0)
Requirement already satisfied: traitlets>=5.4.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (5.7.1)
Requirement already satisfied: tornado>=6.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (6.1)
Requirement already satisfied: matplotlib-inline>=0.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (0.1.6)
Requirement already satisfied: nest-asyncio in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (1.5.6)
```

```
Requirement already satisfied: jupyter-client>=6.1.12 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (7.3.4)
Requirement already satisfied: debugpy>=1.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (1.5.1)
Requirement already satisfied: appnope in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipykernel->jupyter>=0.0->ISLP) (0.1.2)
Requirement already satisfied: widgetsnbextension~=4.0.7 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipywidgets->jupyter>=0.0->ISLP) (4.0.8)
Requirement already satisfied: jupyterlab-widgets~=3.0.7 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipywidgets->jupyter>=0.0->ISLP) (3.0.8)
Requirement already satisfied: prompt-toolkit>=3.0.30 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
console->jupyter>=0.0->ISLP) (3.0.36)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
console->jupyter>=0.0->ISLP) (5.2.0)
Requirement already satisfied: pygments in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
console->jupyter>=0.0->ISLP) (2.11.2)
Requirement already satisfied: pandocfilters>=1.4.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (1.5.0)
Requirement already satisfied: tinycss2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (1.2.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (0.4)
Requirement already satisfied: MarkupSafe>=2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (2.1.1)
Requirement already satisfied: nbformat>=5.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (5.7.0)
Requirement already satisfied: bleach in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (4.1.0)
Requirement already satisfied: jinja2>=3.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (3.1.2)
Requirement already satisfied: beautifulsoup4 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (4.11.1)
```

```
Requirement already satisfied: mistune<2,>=0.8.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (0.5.13)
Requirement already satisfied: jupyterlab-pygments in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (0.1.2)
Requirement already satisfied: defusedxml in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbconvert->jupyter>=0.0->ISLP) (0.7.1)
Requirement already satisfied: ipython-genutils in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (0.2.0)
Requirement already satisfied: prometheus-client in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (0.14.1)
Requirement already satisfied: Send2Trash>=1.8.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (1.8.0)
Requirement already satisfied: terminado>=0.8.3 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (0.17.1)
Requirement already satisfied: argon2-cffi in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (21.3.0)
Requirement already satisfied: nbclassic>=0.4.7 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
notebook->jupyter>=0.0->ISLP) (0.5.2)
Requirement already satisfied: qtpy>=2.0.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
qtconsole->jupyter>=0.0->ISLP) (2.2.0)
Requirement already satisfied: decorator in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (5.1.1)
Requirement already satisfied: jedi>=0.16 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.18.1)
Requirement already satisfied: stack-data in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.2.0)
Requirement already satisfied: pexpect>4.3 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (4.8.0)
Requirement already satisfied: pickleshare in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.7.5)
```

```
Requirement already satisfied: backcall in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.2.0)
Requirement already satisfied: platformdirs>=2.5 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
core!=5.0.*,>=4.12->jupyter-console->jupyter>=0.0->ISLP) (2.5.2)
Requirement already satisfied: notebook-shim>=0.1.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (0.2.2)
Requirement already satisfied: jupyter-server>=1.8 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (1.23.4)
Requirement already satisfied: fastjsonschema in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbformat>=5.1->nbconvert->jupyter>=0.0->ISLP) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
nbformat>=5.1->nbconvert->jupyter>=0.0->ISLP) (4.17.3)
Requirement already satisfied: wcwidth in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from prompt-
toolkit>=3.0.30->jupyter-console->jupyter>=0.0->ISLP) (0.2.5)
Requirement already satisfied: ptyprocess in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
terminado>=0.8.3->notebook->jupyter>=0.0->ISLP) (0.7.0)
Requirement already satisfied: argon2-cffi-bindings in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
argon2-cffi->notebook->jupyter>=0.0->ISLP) (21.2.0)
Requirement already satisfied: soupsieve>1.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
beautifulsoup4->nbconvert->jupyter>=0.0->ISLP) (2.3.2.post1)
Requirement already satisfied: webencodings in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
bleach->nbconvert->jupyter>=0.0->ISLP) (0.5.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
jedi>=0.16->ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.8.3)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter>=0.0->ISLP) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert->jupyter>=0.0->ISLP) (22.1.0)
Requirement already satisfied: websocket-client in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
server>=1.8->nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (0.58.0)
Requirement already satisfied: anyio<4,>=3.1.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from jupyter-
server>=1.8->nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (3.5.0)
```

```
Requirement already satisfied: cffi>=1.0.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from argon2-cffi-
bindings->argon2-cffi->notebook->jupyter>=0.0->ISLP) (1.15.1)
Requirement already satisfied: executing in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.8.3)
Requirement already satisfied: pure-eval in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (0.2.2)
Requirement already satisfied: asttokens in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from stack-
data->ipython>=7.23.1->ipykernel->jupyter>=0.0->ISLP) (2.0.5)
Requirement already satisfied: sniffio>=1.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
anyio<4,>=3.1.0->jupyter-
server>=1.8->nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (1.2.0)
Requirement already satisfied: idna>=2.8 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
anyio<4,>=3.1.0->jupyter-
server>=1.8->nbclassic>=0.4.7->notebook->jupyter>=0.0->ISLP) (3.4)
Requirement already satisfied: pycparser in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook->jupyter>=0.0->ISLP)
(2.21)
```

[98]:
```
!pip install pytorch-lightning
```

```
Requirement already satisfied: pytorch-lightning in
/Users/barnana/anaconda3/lib/python3.10/site-packages (2.0.6)
Requirement already satisfied: typing-extensions>=4.0.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(4.4.0)
Requirement already satisfied: fsspec[http]>2021.06.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(2022.11.0)
Requirement already satisfied: tqdm>=4.57.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(4.64.1)
Requirement already satisfied: lightning-utilities>=0.7.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(0.9.0)
Requirement already satisfied: PyYAML>=5.4 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(6.0)
Requirement already satisfied: torch>=1.11.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(1.12.1)
Requirement already satisfied: torchmetrics>=0.7.0 in
```

```
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(1.0.1)
Requirement already satisfied: numpy>=1.17.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(1.25.1)
Requirement already satisfied: packaging>=17.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from pytorch-lightning)
(22.0)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
fsspec[http]>2021.06.0->pytorch-lightning) (3.8.5)
Requirement already satisfied: requests in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
fsspec[http]>2021.06.0->pytorch-lightning) (2.28.1)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (4.0.2)
Requirement already satisfied: multidict<7.0,>=4.5 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (6.0.4)
Requirement already satisfied: aiosignal>=1.1.2 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (1.3.1)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (2.0.4)
Requirement already satisfied: attrs>=17.3.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (22.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (1.4.0)
Requirement already satisfied: yarl<2.0,>=1.0 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
aiohttp!=4.0.0a0,!=4.0.0a1->fsspec[http]>2021.06.0->pytorch-lightning) (1.9.2)
Requirement already satisfied: certifi>=2017.4.17 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
requests->fsspec[http]>2021.06.0->pytorch-lightning) (2023.5.7)
Requirement already satisfied: idna<4,>=2.5 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
requests->fsspec[http]>2021.06.0->pytorch-lightning) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/Users/barnana/anaconda3/lib/python3.10/site-packages (from
requests->fsspec[http]>2021.06.0->pytorch-lightning) (1.26.14)
```

[99]:
```
from ISLP import load_data
import pandas as pd
```

```
import numpy as np
import seaborn as sns
import math
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, KFold,␣
 ↪cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.cross_decomposition import PLSRegression
import itertools
from sklearn.tree import plot_tree
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=DeprecationWarning)
import torch
import torch.nn as nn
import torch.nn.functional as F
from sklearn.metrics import classification_report
```

[100]:
```
# Creating a function to print output in green and bold
# ANSI escape code for green color and bold font
GREEN_BOLD = '\033[1;32m'

# ANSI escape code to reset colors and font style
RESET = '\033[0m'

def print_green_bold(*args):
    text = ' '.join(str(arg) for arg in args)
    print(GREEN_BOLD + text + RESET)
```

1 Chapter 6

1.1 Question 9

In this exercise, we will predict the number of applications received using the other variables in the College data set. (a) Split the data set into a training set and a test set.

[132]:
```
college_c6q9 = load_data('College')
college_df_c6q9=pd.get_dummies(college_c6q9,columns=['Private'],drop_first=True)

#Create separate dfs for independent and dependent variables
X_c6q9 = college_df_c6q9.drop('Apps', axis=1)
y_c6q9 = college_df_c6q9['Apps']

# Normalize the predictors using StandardScaler
scaler = StandardScaler()
X_normalized_c6q9 = scaler.fit_transform(X_c6q9)

# Split the normalized data into train and test sets
X_train_c6q9, X_test_c6q9, y_train_c6q9, y_test_c6q9 =␣
↪train_test_split(X_normalized_c6q9, y_c6q9, test_size=0.2, random_state=42)
```

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

[133]:
```
# Fit a linear regression model using least squares on the training set
linear_model_c6q9 = LinearRegression()
linear_model_c6q9.fit(X_train_c6q9, y_train_c6q9)

# Predict on the test set
y_pred_c6q9 = linear_model_c6q9.predict(X_test_c6q9)

# Calculate the test error (mean squared error)
test_error_c6q9 = mean_squared_error(y_test_c6q9, y_pred_c6q9)

# Calculate R-squared value
r2_c6q9 = linear_model_c6q9.score(X_test_c6q9, y_test_c6q9)

print_green_bold("Test Error:", test_error_c6q9)
print_green_bold("R-squared:", r2_c6q9)
```

```
Test Error: 1492443.3790390247
R-squared: 0.8877583168400992
```

(c) Fit a ridge regression model on the training set, with chosen by cross-validation. Report the test error obtained.

[134]:
```
# Create the Ridge regression model with automatic alpha selection (RidgeCV)
ridge_cv_c6q9 = RidgeCV(alphas=np.linspace(1,100,10000),␣
 ↪scoring='neg_mean_squared_error')
ridge_cv_c6q9.fit(X_train_c6q9, y_train_c6q9)

# Get the best alpha from RidgeCV
best_alpha_ridge_c6q9 = ridge_cv_c6q9.alpha_

# Predict on the test set
y_pred_ridge_c6q9 = ridge_cv_c6q9.predict(X_test_c6q9)

# Calculate the test error (mean squared error)
test_error_ridge_c6q9 = mean_squared_error(y_test_c6q9, y_pred_ridge_c6q9)

# Calculate R-squared value
r2_ridge_c6q9 = ridge_cv_c6q9.score(X_test_c6q9, y_test_c6q9)

# Print the results
print_green_bold("Best Ridge alpha:", best_alpha_ridge_c6q9)
print_green_bold("Test Error (MSE):", test_error_ridge_c6q9)
print_green_bold("R-squared for Ridge:", r2_ridge_c6q9)
```

```
Best Ridge alpha: 1.0
Test Error (MSE): 1477288.0746996612
R-squared for Ridge: 0.8888980966747928
```

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

[135]:
```
# Create the Lasso regression model with automatic alpha selection
lasso_cv_c6q9 = LassoCV(alphas=np.linspace(1,100,10000))
lasso_cv_c6q9.fit(X_train_c6q9, y_train_c6q9)

# Get the best alpha from LassoCV
best_alpha_lasso_c6q9 = lasso_cv_c6q9.alpha_

# Predict on the test set
y_pred_lasso_c6q9 = lasso_cv_c6q9.predict(X_test_c6q9)

# Calculate the test error (mean squared error)
test_error_lasso_c6q9 = mean_squared_error(y_test_c6q9, y_pred_lasso_c6q9)

# Get the number of non-zero coefficient estimates
non_zero_coefficients_c6q9 = (lasso_cv_c6q9.coef_ != 0).sum()

# Print the results
print_green_bold("Best Lasso alpha:", best_alpha_lasso_c6q9)
print_green_bold("Test Error (MSE):", test_error_lasso_c6q9)
```

```
print_green_bold("Number of non-zero coefficients:", non_zero_coefficients_c6q9)
```

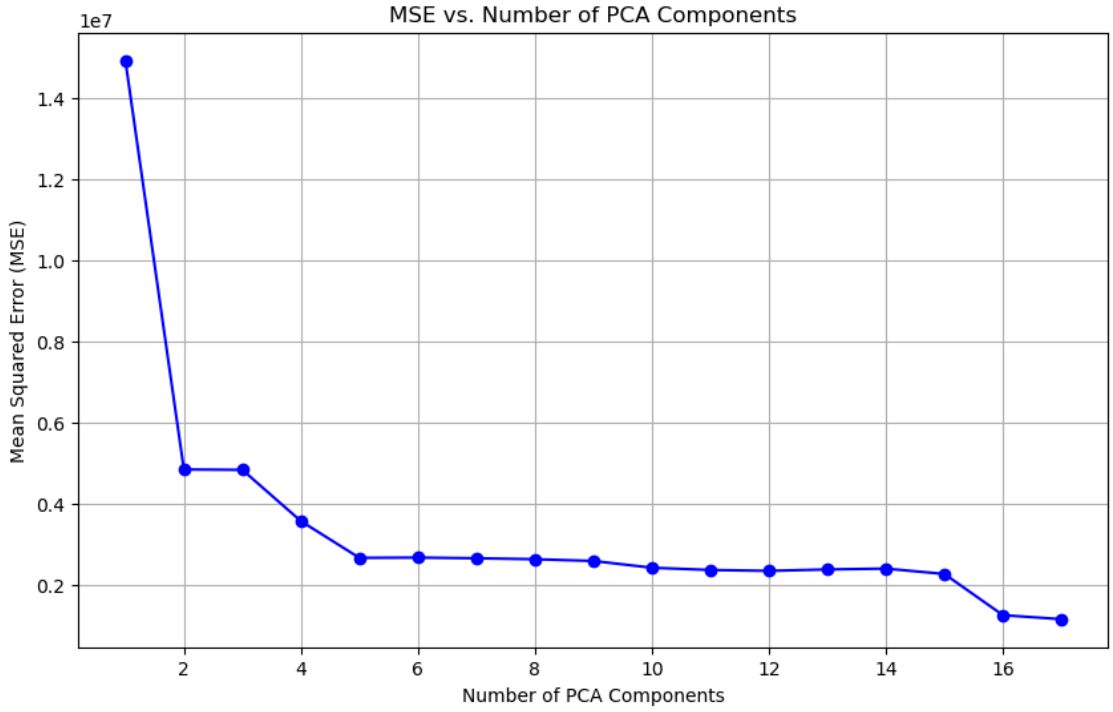
```
Best Lasso alpha: 1.00990099009901
Test Error (MSE): 1487932.4877020943
Number of non-zero coefficients: 17
```

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

[136]:
```
pca_c6q9 = PCA()
pipe_pca_c6q9 = Pipeline([('pca', pca_c6q9),
    ('linear_model', linear_model_c6q9)
])

# Define the parameter grid for the grid search
param_grid_c6q9 = {'pca__n_components': range(1, 18)}

# Create the k-fold cross-validator
kfold_c6q9 = KFold(n_splits=5, random_state=42, shuffle=True)

# Perform grid search using cross-validation
grid_pca_c6q9 = GridSearchCV(pipe_pca_c6q9, param_grid_c6q9, cv=kfold_c6q9,␣
 ↪scoring='neg_mean_squared_error')
grid_pca_c6q9.fit(X_train_c6q9, y_train_c6q9)

# Get the best model from the grid search
best_model_pca_c6q9 = grid_pca_c6q9.best_estimator_

# Predict on the test set
y_pred_pca_c6q9 = best_model_pca_c6q9.predict(X_test_c6q9)

# Calculate the test error (mean squared error)
test_error_pca_c6q9 = mean_squared_error(y_test_c6q9, y_pred_pca_c6q9)

# Access the coefficients of the linear regression model in the best model
coefficients_pca_c6q9 = best_model_pca_c6q9.named_steps['linear_model'].coef_

# Print the results
print_green_bold("Best PCA n_components:", grid_pca_c6q9.
 ↪best_params_['pca__n_components'])
print_green_bold("Test Error:", test_error_pca_c6q9)
print_green_bold("Coefficients:", coefficients_pca_c6q9)

# Get the grid search results
results_pca_c6q9 = pd.DataFrame(grid_pca_c6q9.cv_results_)

# Plot n_components against MSE
plt.figure(figsize=(10, 6))
```

```
plt.plot(results_pca_c6q9['param_pca__n_components'],␣
  ↪-results_pca_c6q9['mean_test_score'], marker='o', linestyle='-', color='b')
plt.xlabel('Number of PCA Components')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('MSE vs. Number of PCA Components')
plt.grid(True)
plt.show()
```

```
Best PCA n_components: 17
Test Error: 1492443.3790390252
Coefficients: [ 362.0663594  1604.09202573  192.60792738 1302.29478033

500.66544951

  -88.96016676 -186.56511228   -4.8955025  -261.41109187 -802.65953031

 -110.96924027 -317.74716208  184.51710541   51.04713704 2158.45463654

 2494.26794009 1542.08876475]
```

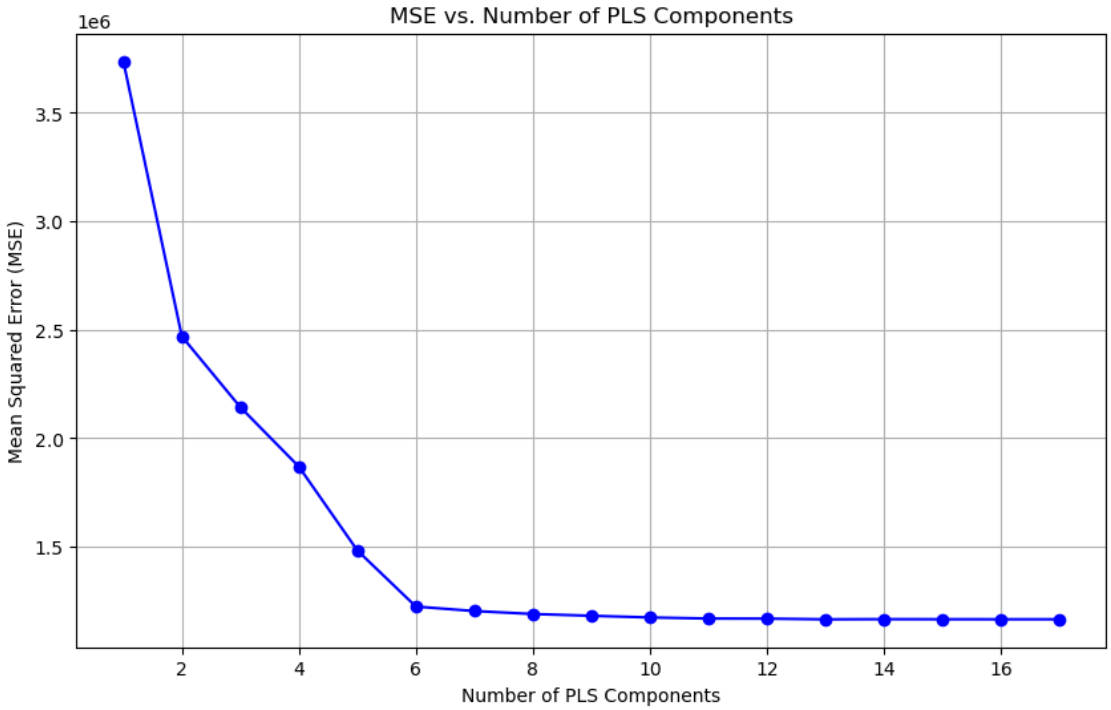


Note that while the MSE is the lowest at 17 components, the largest drop in MSE happens by the time we have 6 components. Post that, the MSE falls gradually till 15 and then there is a sharper dip. In a real world scenario, we can use this figure to pick an appropriate number of components that is less than 17 if our goal is dimensionality reduction.

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

[137]:
```
# Create the PLS regression model
pls_c6q9 = PLSRegression()

# Define the parameter grid for the grid search
param_grid_c6q9 = {'n_components': range(1, 18)}

# Create the k-fold cross-validator
kfold_c6q9 = KFold(n_splits=5, random_state=42, shuffle=True)

# Perform grid search using cross-validation
grid_pls_c6q9 = GridSearchCV(pls_c6q9, param_grid_c6q9, cv=kfold_c6q9,␣
 ↪scoring='neg_mean_squared_error')
grid_pls_c6q9.fit(X_train_c6q9, y_train_c6q9)

# Get the best model from the grid search
best_model_pls_c6q9 = grid_pls_c6q9.best_estimator_

# Predict on the test set
y_pred_pls_c6q9 = best_model_pls_c6q9.predict(X_test_c6q9)

# Calculate the test error (mean squared error)
test_error_pls_c6q9 = mean_squared_error(y_test_c6q9, y_pred_pls_c6q9)

# Print the results
print_green_bold("Best PLS n_components:", grid_pls_c6q9.
 ↪best_params_['n_components'])
print_green_bold("Test Error (MSE):", test_error_pls_c6q9)

# Get the grid search results
results_pls_c6q9 = pd.DataFrame(grid_pls_c6q9.cv_results_)

# Plot n_components against MSE
plt.figure(figsize=(10, 6))
plt.plot(results_pls_c6q9['param_n_components'],␣
 ↪-results_pls_c6q9['mean_test_score'], marker='o', linestyle='-', color='b')
plt.xlabel('Number of PLS Components')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('MSE vs. Number of PLS Components')
plt.grid(True)
plt.show()
```

```
Best PLS n_components: 13
Test Error (MSE): 1483820.2818416457
```



Again, note that while the MSE is the lowest at 13 components, there is very little reduction in MSE after we reach 6 components.

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

The primary takeaways from the analysis are as follows :

- Ridge Regression achieved the lowest test error (MSE) among the tested models
- Lasso Regression did not reduce features, and its performance was slightly worse than Ridge Regression
- Principal Component Analysis (PCA) and Linear Regression performed similarly and did not improve prediction performance significantly
- The PCA model also failed to reduce dimensions as the number of components chosen was equal to the number of predictors
- Partial Least Squares (PLS) Regression had an intermediate test error but did not outperform Ridge Regression or Lasso Regression
- PLS, however, managed to reduce the number of components to 13

When it comes to dimensionality reduction, it is essential to look at the graph of MSE vs Number of components before deciding how many components to choose. Blindly following the model results and picking the number of components with the lowest MSE might return suboptimal results with regards to dimensionality reduction.

In conclusion, all 5 models used have similar values for test error. If we were to pick our best performing model i.e. the Ridge regression, we will be able to predict the number of college applications

reasonably well with an R quared value of 89%.

1.2 Question 11

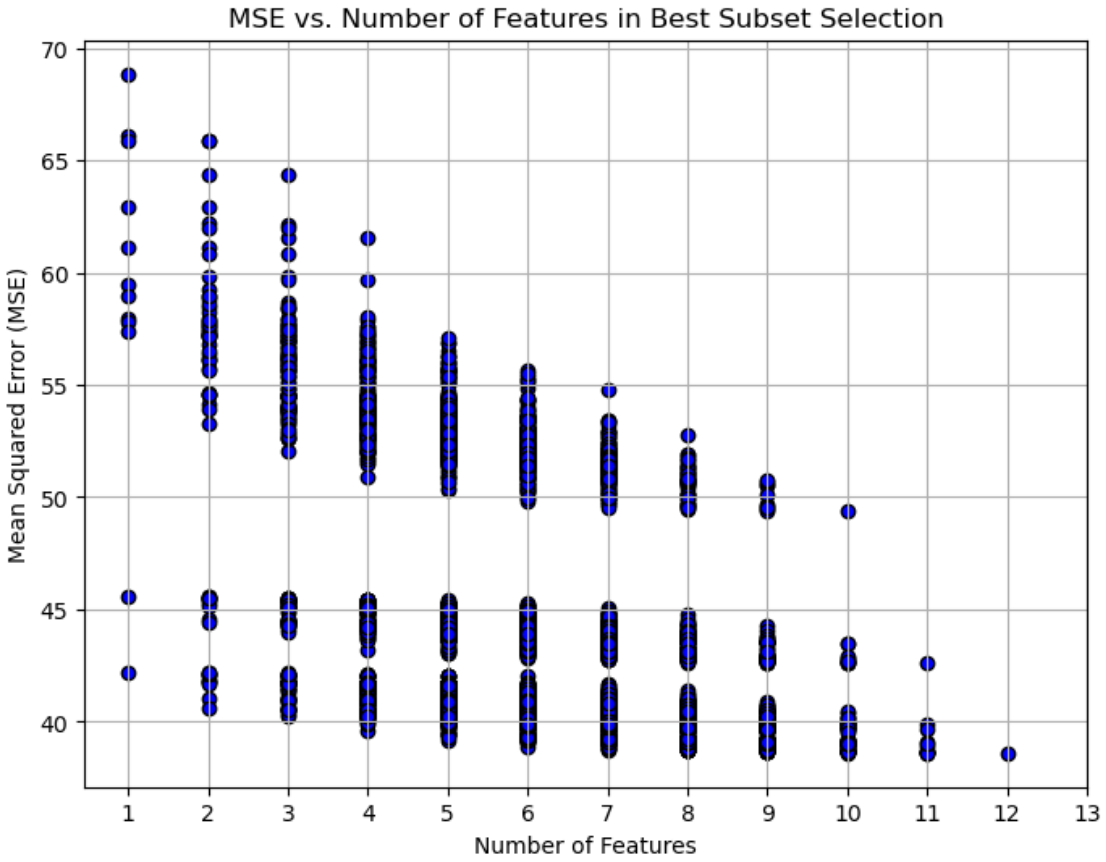
We will now try to predict per capita crime rate in the Boston data set. (a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

1.2.1 Best Subset selection

[138]:
```
boston_c6q11 = load_data('Boston')

# Separate the target variable (crim) and the predictor variables (features)
X_c6q11 = boston_c6q11.drop(columns=['crim'], axis=1)
y_c6q11 = boston_c6q11['crim']

# Normalize the predictors using StandardScaler
scaler = StandardScaler()
X_normalized_c6q11 = scaler.fit_transform(X_c6q11)

# Convert X_normalized back to DataFrame with column names
X_normalized_df_c6q11 = pd.DataFrame(X_normalized_c6q11, columns=X_c6q11.
 ↪columns)

# Split the normalized data into train and test sets
X_train_c6q11, X_test_c6q11, y_train_c6q11, y_test_c6q11 =␣
 ↪train_test_split(X_normalized_df_c6q11, y_c6q11, test_size=0.
 ↪3,random_state=42)


# 1. Best Subset Selection
def best_subset_selection(X, y, max_features):
    best_subset_model = None
    best_subset_score = float('inf')  # Set an initial high value for comparison
    mse_values = []

    for k in range(1, max_features + 1):
        for subset in itertools.combinations(X.columns, k):
            subset_X = X[list(subset)]
            model = LinearRegression()
            model.fit(subset_X, y)
            y_pred = model.predict(subset_X)
            mse = mean_squared_error(y, y_pred)
            mse_values.append((k, mse))

            if mse < best_subset_score:
                best_subset_score = mse
```

```
                best_subset_model = model
                best_subset_features = subset

    return best_subset_model, best_subset_features, mse_values

max_features_c6q11 = 13
best_model_c6q11, best_features_c6q11, mse_values_c6q11 =␣
 ↪best_subset_selection(X_train_c6q11, y_train_c6q11, max_features_c6q11)

print_green_bold("Best Subset Selection - Selected Features:",␣
 ↪best_features_c6q11)

# Print MSE for the best model on the test set
best_model_subset_X_test_c6q11 = X_test_c6q11[list(best_features_c6q11)]
best_model_y_pred_c6q11 = best_model_c6q11.
 ↪predict(best_model_subset_X_test_c6q11)
best_model_mse_c6q11 = mean_squared_error(y_test_c6q11, best_model_y_pred_c6q11)
print_green_bold("MSE for Best Subset Selection Model on Test Set:",␣
 ↪best_model_mse_c6q11)

# Plot the MSE vs. Number of Features using a scatter plot
plt.figure(figsize=(8, 6))
mse_values_c6q11.sort(key=lambda x: x[0])  # Sort by number of features (k) for␣
 ↪plotting
num_features_c6q11, mse_c6q11 = zip(*mse_values_c6q11)
plt.scatter(num_features_c6q11, mse_c6q11, marker='o', color='blue',␣
 ↪edgecolors='black')
plt.xlabel("Number of Features")
plt.ylabel("Mean Squared Error (MSE)")
plt.title("MSE vs. Number of Features in Best Subset Selection")
plt.xticks(np.arange(1, max_features_c6q11 + 1))
plt.grid(True)
plt.show()
```

```
Best Subset Selection - Selected Features: ('zn', 'indus', 'chas', 'nox',
'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'lstat', 'medv')
MSE for Best Subset Selection Model on Test Set: 46.732462754277456
```



MSE is lowest when number of features is equal to 12.

1.2.2 Lasso

[139]:
```
# 2. Lasso Regression with Cross-validation (LassoCV)
lasso_cv_model_c6q11 = LassoCV(alphas=np.linspace(1,100,10000))
lasso_cv_model_c6q11.fit(X_train_c6q11, y_train_c6q11)
lasso_cv_alpha_c6q11 = lasso_cv_model_c6q11.alpha_
lasso_y_pred_c6q11 = lasso_cv_model_c6q11.predict(X_test_c6q11)
lasso_mse_c6q11 = mean_squared_error(y_test_c6q11, lasso_y_pred_c6q11)

print_green_bold("LassoCV Regression - Alpha:", lasso_cv_alpha_c6q11)
print_green_bold("LassoCV Regression - MSE:", lasso_mse_c6q11)
```

```
LassoCV Regression - Alpha: 1.0
LassoCV Regression - MSE: 51.76637176281037
```

1.2.3 Ridge

[140]:
```
# 3. Ridge Regression with Cross-validation (RidgeCV)
ridge_cv_model_c6q11 = RidgeCV(alphas=np.linspace(1,100,10000))  # alphas for␣
 ↪cross-validation
ridge_cv_model_c6q11.fit(X_train_c6q11, y_train_c6q11)
ridge_cv_alpha_c6q11 = ridge_cv_model_c6q11.alpha_
ridge_y_pred_c6q11 = ridge_cv_model_c6q11.predict(X_test_c6q11)
ridge_mse_c6q11 = mean_squared_error(y_test_c6q11, ridge_y_pred_c6q11)

print_green_bold("RidgeCV Regression - Alpha:", ridge_cv_alpha_c6q11)
print_green_bold("RidgeCV Regression - MSE:", ridge_mse_c6q11)
```

```
RidgeCV Regression - Alpha: 6.425742574257426
RidgeCV Regression - MSE: 46.7750692659645
```

1.2.4 PCA

[141]:
```
# 4. Principal Component Regression (PCR)
pca_c6q11 = PCA(n_components=5)
X_train_pca_c6q11 = pca_c6q11.fit_transform(X_train_c6q11)
X_test_pca_c6q11 = pca_c6q11.transform(X_test_c6q11)

pca_model_c6q11 = LinearRegression()
pca_model_c6q11.fit(X_train_pca_c6q11, y_train_c6q11)
pca_y_pred_c6q11 = pca_model_c6q11.predict(X_test_pca_c6q11)
pca_mse_c6q11 = mean_squared_error(y_test_c6q11, pca_y_pred_c6q11)

print_green_bold("Principal Components Regression - MSE:", pca_mse_c6q11)
```

```
Principal Components Regression - MSE: 51.17579244921238
```

Comparison Summary: - Best Subset Selection and RidgeCV Regression performed similarly and achieved the lowest MSE values among the approaches. - LassoCV Regression and PCR had slightly higher MSE values, suggesting relatively less predictive accuracy. - Best Subset Selection's direct feature selection and RidgeCV's ability to handle multicollinearity might have contributed to their competitive performances.

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross- validation, or some other reasonable alternative, as opposed to using training error.

Note that we are evaluating models based on test error not training error.

I would pick the Ridge Regression as the best model for the following reasons: - Based on the output provided, the Ridge regression model seems to perform the best on this dataset. This is justified by it having the lowest mean squared error (MSE) on the test set compared to the other models.

- While the Best Subset Selection model has a slightly lower MSE, it uses all but one predictor,

which can lead to overfitting and poor generalization to unseen data. On the other hand, Ridge Regression uses all predictors but shrinks their coefficients, which can improve model generalizability.

- The Lasso regression model can perform both variable selection and regularization (which helps to manage multicollinearity and overfitting) but has a higher MSE.
- Although Principal Component Regression is a good model to consider if the predictors are highly correlated (as it removes the multicollinearity through PCA before regression) in this case, it appears that the other models perform better.

(c) Does your chosen model involve all of the features in the data set? Why or why not?

Ridge Regression model, which had the best performance based on the provided output, does involve all the features in the dataset. The Ridge Regression model tries to balance two things - Minimizing the prediction error on the training data and minimizing the size of the regression coefficients. This second part is the regularization aspect. This process can help prevent overfitting, where the model is too complex and performs well on the training data but poorly on unseen data. It does this by adding a degree of bias to the model, with the trade-off being a reduction in variance. It is important to note that while Ridge Regression does involve all features, it doesn't treat all features equally. Instead, it reduces the coefficients of less important features, effectively reducing their impact on the model's predictions.

[ ]: