Module 3
Lesson 1 - What Is Lasso Regression?

(c) Wesleyan University, 2016

Lasso Regression

- *Penalized regression method*

Lasso regression is what is called
the Penalized regression method,

Lasso Regression

- *Penalized regression method*

often used in machine learning to
select the subset of variables.

Lasso Regression

- Penalized regression method
- Supervised

- Shrinkage
- Variable Selection

Least
Absolute
Selection
Shrinkage
Operator

Shrinkage & Selection

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + e$$

The LASSO imposes a constraint on the sum of the absolute values of the model

Shrinkage & Selection

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + e$$

parameters, where the sum has
a specified constant as an upper bound.

Shrinkage & Selection

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + e$$

some variables to shrink towards zero.

Shrinkage & Selection

$$Y = \beta_0 + \beta_1 x_1 + 0 + 0 + \beta_4 x_4 + 0 + \beta_6 x_6 + e$$

Shrinkage & Selection

Shrinkage

Constraint on parameters that shrinks coefficients toward zero

the variables most strongly associated with the target corresponds variable.

Shrinkage & Selection

Shrinkage Constraint on parameters that shrinks coefficients toward zero

Selection Identifies the most important variables associated with the response variable

That is the variable selection process.

Shrinkage & Selection

Shrinkage Constraint on parameters that shrinks coefficients toward zero

Selection Identifies the most important variables associated with the response variable

It goes to obtain the subset of predictors that minimizes prediction error.

Why Use Lasso Regression?

- Greater prediction accuracy

the predictors is approximately linear and
you have a large number of observations,

Why Use Lasso Regression?

- Greater prediction accuracy

then OLS regression parameter estimates will have low bias and low variance.

Why Use Lasso Regression?

- Greater prediction accuracy

However, if you have a relatively small number of observations and a large number

Why Use Lasso Regression?

- Greater prediction accuracy

of predictors, then the variance of the OLS perimeter estimates will be higher.

Why Use Lasso Regression?

- Greater prediction accuracy

In this case, Lasso Regression is useful because shrinking the regression

Why Use Lasso Regression?

- Greater prediction accuracy

coefficient can reduce variance without
a substantial increase in bias.

Why Use Lasso Regression?

- Greater prediction accuracy
- Increase model interpretability

produces a simpler model that selects only the most important predictors.

Why Use Lasso Regression?

Tuning parameter: lambda (λ)

In Lasso Regression,
a tuning parameter called lambda

Why Use Lasso Regression?

Tuning parameter: lambda (λ)

As lambda increases, more coefficients are reduced to zero that is fewer

Why Use Lasso Regression?

Tuning parameter: lambda (λ)

- When $\lambda=0$ then it's OLS regression

With Lasso Regression where lambda is equal to zero then we have an OLS

Response (Target) Variable:

SCHCONN1 (School Connectedness; range 6-38)

Predictors (Categorical)

MALE (1=male 0=female)
HISPANIC (1=yes, 0=no)
WHITE (1=yes, 0=no)
BLACK (1=yes, 0=no)
NAMERICAN (1=yes, 0=no)
ASIAN (1=yes, 0=no)
ALCEVER1 (1=yes, 0=no)
MAREVER1 (1=yes, 0=no)
COCEVER1 (1=yes, 0=no)
INHEVER1 (1=yes, 0=no)
CIGAVAIL (1=yes, 0=no)
PASSIST (1=yes, 0=no)
EXPEL1 (1=yes, 0=no)

Predictors (Quantitative)

AGE
ALCPROBS1 (alcohol problems 0-6)
DEVIANT1 (deviant behavior scale)
VIOL1 (violent behavior scale)
DEP1 (depression scale)
ESTEEM1 (self esteem scale)
PARPRES (parental presence scale)
PARACTV (parent activities scale)
FAMCONCT (family connectedness scale)
GPA1 (Grade Point Average; 4.0 scale)

Response (Target) Variable:

SCHCONN1 (School Connectedness; range 6-38)

Predictors (Categorical)

MALE (1=male 0=female)
HISPANIC (1=yes, 0=no)
WHITE (1=yes, 0=no)
BLACK (1=yes, 0=no)
NAMERICAN (1=yes, 0=no)
ASIAN (1=yes, 0=no)
ALCEVER1 (1=yes, 0=no)
MAREVER1 (1=yes, 0=no)
COCEVER1 (1=yes, 0=no)
INHEVER1 (1=yes, 0=no)
CIGAVAIL (1=yes, 0=no)
PASSIST (1=yes, 0=no)
EXPEL1 (1=yes, 0=no)

Predictors (Quantitative)

AGE
ALCPROBS1 (alcohol problems 0-6)
DEVIANT1 (deviant behavior scale)
VIOL1 (violent behavior scale)
DEP1 (depression scale)
ESTEEM1 (self esteem scale)
PARPRES (parental presence scale)
PARACTV (parent activities scale)
FAMCONCT (family connectedness scale)
GPA1 (Grade Point Average; 4.0 scale)

This is a pretty large number of predictor variables, so

Response (Target) Variable:

SCHCONN1 (School Connectedness; range 6-38)

Predictors (Categorical)

MALE (1=male 0=female)
HISPANIC (1=yes, 0=no)
WHITE (1=yes, 0=no)
BLACK (1=yes, 0=no)
NAMERICAN (1=yes, 0=no)
ASIAN (1=yes, 0=no)
ALCEVER1 (1=yes, 0=no)
MAREVER1 (1=yes, 0=no)
COCEVER1 (1=yes, 0=no)
INHEVER1 (1=yes, 0=no)
CIGAVAIL (1=yes, 0=no)
PASSIST (1=yes, 0=no)
EXPEL1 (1=yes, 0=no)

Predictors (Quantitative)

AGE
ALCPROBS1 (alcohol problems 0-6)
DEVIANT1 (deviant behavior scale)
VIOL1 (violent behavior scale)
DEP1 (depression scale)
ESTEEM1 (self esteem scale)
PARPRES (parental presence scale)
PARACTV (parent activities scale)
FAMCONCT (family connectedness scale)
GPA1 (Grade Point Average; 4.0 scale)

using OLS multiple regression analysis would not be ideal,

Response (Target) Variable:

SCHCONN1 (School Connectedness; range 6-38)

Predictors (Categorical)

MALE (1=male 0=female)
HISPANIC (1=yes, 0=no)
WHITE (1=yes, 0=no)
BLACK (1=yes, 0=no)
NAMERICAN (1=yes, 0=no)
ASIAN (1=yes, 0=no)
ALCEVER1 (1=yes, 0=no)
MAREVER1 (1=yes, 0=no)
COCEVER1 (1=yes, 0=no)
INHEVER1 (1=yes, 0=no)
CIGAVAIL (1=yes, 0=no)
PASSIST (1=yes, 0=no)
EXPEL1 (1=yes, 0=no)

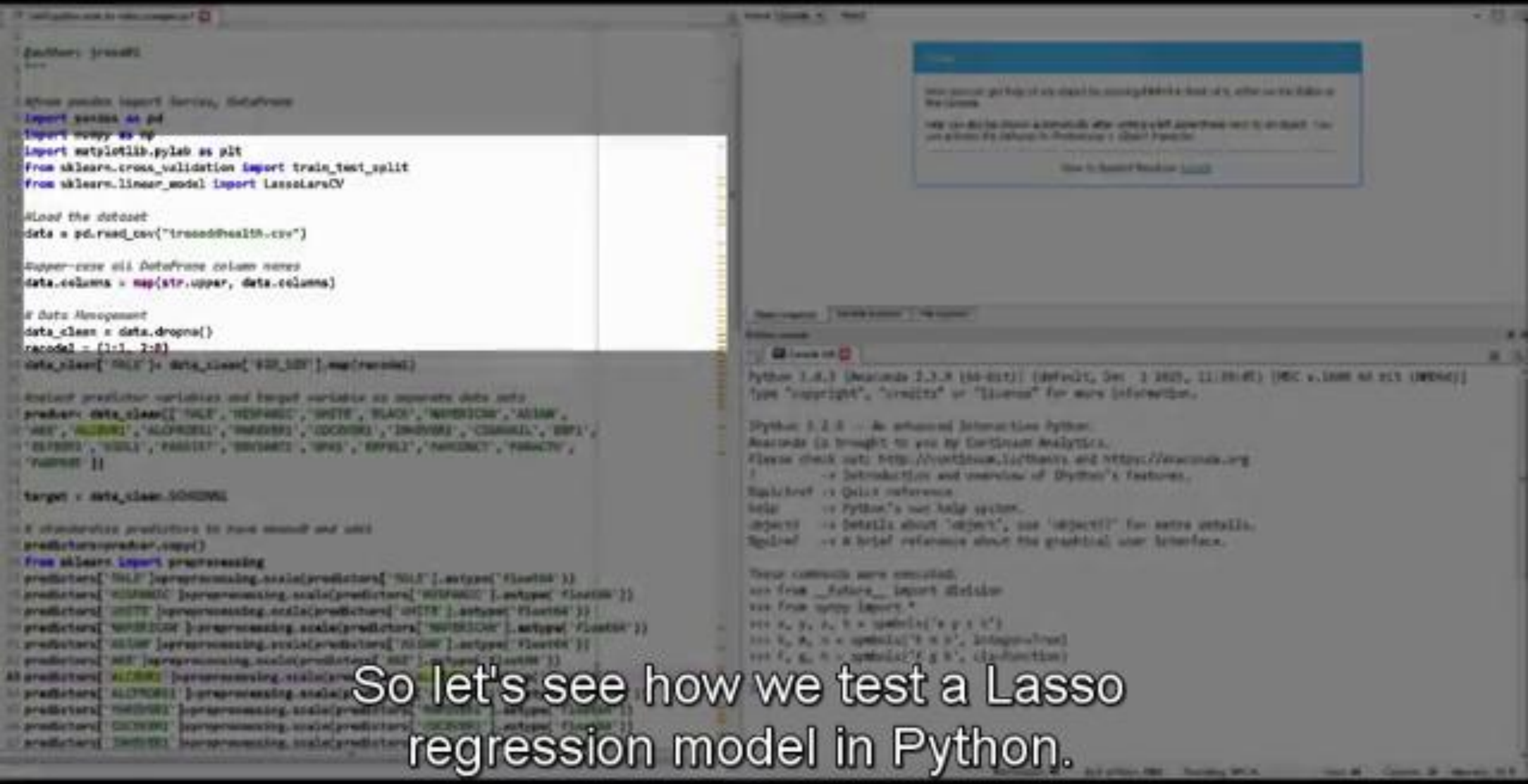
Predictors (Quantitative)

AGE
ALCPROBS1 (alcohol problems 0-6)
DEVIANT1 (deviant behavior scale)
VIOL1 (violent behavior scale)
DEP1 (depression scale)
ESTEEM1 (self esteem scale)
PARPRES (parental presence scale)
PARACTV (parent activities scale)
FAMCONCT (family connectedness scale)
GPA1 (Grade Point Average; 4.0 scale)

of these predictors that most accurately predicts school connectedness.

Module 3
Lesson 2a - Data Management for Lasso Regression in Python

(c) Wesleyan University, 2016


import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV
data = pd.read_csv("treasdhealth.csv")
data.columns = map(str.upper, data.columns)
data_clean = data.dropna()
target = data_clean.SCHIZREC
So let's see how we test a Lasso regression model in Python.

```
@author: jrosa01

#from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV

#Load the dataset
data = pd.read_csv("tree4dHealth.csv")

#upper-case all DataFrame column names
data.columns = map(str.upper, data.columns)

# Data Management
data_clean = data.dropna()
recode1 = {1:1, 2:0}
data_clean['MALE']= data_clean['BIO_SEX'].map(recode1)

#select predictor variables and target variable as separate data sets
predvar= data_clean[['MALE','HISPANIC','WHITE','BLACK','NAMERICAN','ASIAN',
'AGE','ALCEVR1','ALCPROB1','MARIEVR1','COCEVR1',
'ESTEEM1','VIOL1','PASSIST','DEVIANT1','GPA1',
'PARPRES']]

target = data_clean.SCHCONN1
```

First, I will call in
the libraries that I will need.

```
#from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV
```

```
#Load the dataset
data = pd.read_csv("treasdhealth.csv")

#upper-case all DataFrame column names
data.columns = map(str.upper, data.columns)

# Data Management
data_clean = data.dropna()
recode1 = {1:1, 2:0}
data_clean['MALE']= data_clean['BIO_SEX'].map(recode1)

#select predictor variables and target variable as separate data sets
predvar= data_clean[['MALE','HISPANIC','WHITE','BLACK','NAMERICAN','ASIAN',
'AGE','ALCEVR1','ALCPROB1',
'ESTEEM1','VIOL1','PASSIST',
'PARPRES']]

target = data_clean.SOHCONN1
```

the train_test_split function from the sklearn.cross_validation library,

```python
#from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV
```

```python
#Load the dataset
data = pd.read_csv("treeaddhealth.csv")

#Upper-case all DataFrame column names
data.columns = map(str.upper, data.columns)

# Data Management
data_clean = data.dropna()
recode1 = {1:1, 2:0}
data_clean['MALE']= data_clean['BIO_SEX'].map(recode1)

#select predictor variables and target variable as separate data sets
predvar= data_clean[['MALE','HISPANIC','WHITE','BLACK','AMERICAN','ASIAN',
'AGE','ALCEVR1','ALCPROBS1',
'ESTEEM1','VIOL1','PASSIST','DEVIANT1','GPA2','EXPELL1','PARCONCT','PARACTV',
'PARPRES']]

target = data_clean.SCHCONN1
```

and the LassoLarsCV function from the sklearn.linear_model library.

```
@author: jrose01
"""

#from pandas import Series, Dataframe
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV

#Load the dataset
data = pd.read_csv("treasdhealth.csv")

#upper-case all DataFrame column names
data.columns = map(str.upper, data.columns)

# Data Management
data_clean = data.dropna()
recode1 = {1:1, 2:0}
data_clean['MALE']= data_clean['BIO_SEX'].map(recode1)

#select predictor variables and target variable as separate data sets
predvar= data_clean[['MALE','HISPANIC','WHITE','BLACK','NAMERICAN','ASIAN',
'AGE','ALCEVR1','ALCPD
'ESTEEM1','VIOL1','PASSIV
'PARPRES']]

target = data_clean.SCHCONN1
```

Namely, I want to create a new dataset called data_clean in which I will delete

```python
target = data_clean.SCHCONN1

# standardize predictors to have mean=0 and sd=1
predictors=predvar.copy()
from sklearn import preprocessing
predictors['MALE']=preprocessing.scale(predictors['MALE'].astype('float64'))
predictors['HISPANIC']=preprocessing.scale(predictors['HISPANIC'].astype('float64'))
predictors['WHITE']=preprocessing.scale(predictors['WHITE'].astype('float64'))
predictors['NAMERICAN']=preprocessing.scale(predictors['NAMERICAN'].astype('float64'))
predictors['ASIAN']=preprocessing.scale(predictors['ASIAN'].astype('float64'))
predictors['AGE']=preprocessing.scale(predictors['AGE'].astype('float64'))
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))
```

To standardize the predictors,

Object inspector | Variable

IPython console

Console 1/A

```
...: data.col
...:
...: # Data
...: data_cle
...: recode1
...: data_cle
...:
...: #select
...: predvar
...: 'AGE','A
...: 'ESTEEM
...: 'PARPRES
...:
...: target =
C:\Users\jrose01
packages\spyder
```

```
target = data_clean.SCHCONN1

# standardize predictors to have mean=0 and sd=1
predictors=predvar.copy()
from sklearn import preprocessing
predictors['MALE']=preprocessing.scale(predictors['MALE'].astype('float64'))
predictors['HISPANIC']=preprocessing.scale(predictors['HISPANIC'].astype('float64'))
predictors['WHITE']=preprocessing.scale(predictors['WHITE'].astype('float64'))
predictors['NAMERICAN']=preprocessing.scale(predictors['NAMERICAN'].astype('float64'))
predictors['ASIAN']=preprocessing.scale(predictors['ASIAN'].astype('float64'))
predictors['AGE']=preprocessing.scale(predictors['AGE'].astype('float64'))
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=p
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARAC
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))
```

I'm going to first create a copy of my predvar data frame and name it predictors.

Object inspector | Variabl

IPython console

Console 1/A

```
...: data.col
...:
...: # Data
...: data_cle
...: recode1
...: data_cle
...:
...: #select
...: predvar
...: 'AGE','A
...: 'ESTEEM
...: 'PARPRES
...:
...: target =
C:\Users\jrose01
packages\spyder
```

```
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROB51']=preprocessing.scale(predictors['ALCPROB51'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))
```

First, I list the two training data sets.

```
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROB1']=preprocessing.scale(predictors['ALCPROB1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=True,).fit(pred_train,tar_train);

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))
```

The random_state option specifies a random number seed to ensure that the data

```
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROB1']=preprocessing.scale(predictors['ALCPROB1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regr
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))
```

are randomly split the same
way if I run the code again.

Module 3
Lesson 2b - Testing a Lasso Regression Model in Python

(c) Wesleyan University, 2016

```python
predictors=predvar.copy()
from sklearn import preprocessing
predictors['MALE']=preprocessing.scale(predictors['MALE'].astype('float64'))
predictors['HISPANIC']=preprocessing.scale(predictors['HISPANIC'].astype('float64'))
predictors['WHITE']=preprocessing.scale(predictors['WHITE'].astype('float64'))
predictors['NAMERICAN']=preprocessing.scale(predictors['NAMERICAN'].astype('float64'))
predictors['ASIAN']=preprocessing.scale(predictors['ASIAN'].astype('float64'))
predictors['AGE']=preprocessing.scale(predictors['AGE'].astype('float64'))
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROB1']=preprocessing.scale(predictors['ALCPROB1'].astype('float64'))
predictors['MAREVR1']=preprocessing.scale(predictors['MAREVR1'].astype('float64'))
predictors['COCEVR1']=preprocessing.scale(predictors['COCEVR1'].astype('float64'))
predictors['INHEVR1']=preprocessing.scale(predictors['INHEVR1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')
```

using the LASSO LarsCV function from the sklearn linear model library.

```
predictors=predvar.copy()
from sklearn import preprocessing
predictors['MALE']=preprocessing.scale(predictors['MALE'].astype('float64'))
predictors['HISPANIC']=preprocessing.scale(predictors['HISPANIC'].astype('float64'))
predictors['WHITE']=preprocessing.scale(predictors['WHITE'].astype('float64'))
predictors['NAMERICAN']=preprocessing.scale(predictors['NAMERICAN'].astype('float64'))
predictors['ASIAN']=preprocessing.scale(predictors['ASIAN'].astype('float64'))
predictors['AGE']=preprocessing.scale(predictors['AGE'].astype('float64'))
predictors['ALCEVR1']=preprocessing.scale(predictors['ALCEVR1'].astype('float64'))
predictors['ALCPROB1']=preprocessing.scale(predictors['ALCPROB1'].astype('float64'))
predictors['MARIJEVR1']=preprocessing.scale(predictors['MARIJEVR1'].astype('float64'))
predictors['COCAINEVR1']=preprocessing.scale(predictors['COCAINEVR1'].astype('float64'))
predictors['INHEVR1']=preprocessing.scale(predictors['INHEVR1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='dashed',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progr
```

LASSO Regression has a couple of different model selection algorithms.

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O

This algorithm starts with no predictors in the model and

L A R Algorithm

L	A	R
E	N	E
A	G	G
S	L	R
T	E	E
		S
		S
		I
		O
		N

adds a predictor at each step.

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O

It first adds a predictor that is most correlated with the response variable and

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O

moves it towards least score estimate
until there is another predictor.

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O
N

That is equally correlated
with the model residual.

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O
N

It adds this predictor to the model and

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O

starts the least square estimation
process over again with both variables.

L A R Algorithm

E N E
A G G
S L R
T E E
S
S
I
O

The LAR algorithm continues with
this process until it has tested

L A R Algorithm

L	A	R
E	N	E
A	G	G
S	L	R
T	E	E
		S
		S
		I
		O
		N

all the predictors.

L A R Algorithm

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + e$$

Parameter estimates at any step are shrunk
and predictors with coefficients that have

L A R Algorithm

$$Y = \beta_0 + \beta_1 x_1 + \qquad \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + e$$

shrunk to zero are removed from the model
and the process starts all over again.

```
predictors['ALCPROB51']=preprocessing.scale(predictors['ALCPROB51'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

which asks Python to use k-fold cross-validation with ten

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.co

# plot coefficien
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

The option precompute=False tells Python not to use a precomputed matrix.

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns,

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

This option would be helpful
with very large models,

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.18
 'BLACK': -0.6
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.85
 'DEVIANT1':
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.col

# plot coefficient progressio
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

because the precomputed matrix can speed up calculations.

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.column

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

However, the model I'm going to test is not very large.

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.colu

# plot coefficient
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

To fit the Lasso regression on the training set after the parenthesis, I add

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

So what I am doing here is using k-fold cross-validation in

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.colu

# plot coefficient p
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

which the first fold of the training data set is treated as a validation set,

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns,

# plot coefficient progre
m_log_alphas = -np.log10(
ax = plt.gca()
```

and the model is estimated
using the remaining nine folds.

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROB51']=preprocessing.scale(predictors['ALCPROB51'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

At each step of the estimation process,

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROB51':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.

# plot coefficient
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

when a new predictor is entered into the model, the mean-square error for

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictor

# plot coefficie
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

the validation fold is calculated for each of the other nine folds and then averaged.

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROB51']=preprocessing.scale(predictors['ALCPROB51'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.co

# plot coefficien
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

The model that produces the lowest mean-square error is selected by Python

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROB51':
 'ASIAN': 0.1E
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

The first thing we can do is tell Python to print the variables, and

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.18
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

the regression coefficients from the model that was retained

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.18
 'BLACK': -0.6
 'CIGAVAIL': -
 'COCEVER1': -
 'DEP1': -0.85
 'DEVIANT1': -
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.colum

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

by the model selection process
as the best fitting model.

Out[5]:
{'AGE': 0.2173
'ALCEVR1': -0
'ALCPROBS1':
'ASIAN': 0.18
'BLACK': -0.6
'CIGAVAIL': -
'COCEVER1': -
'DEP1': -0.85
'DEVIANT1': -
'ESTEEM1': 1.
'EXPEL1': -0.
'FAMCONCT': 0
'GPA1': 0.665
'HISPANIC': 0
'INHEVER1': 0
'MALE': -0.21

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.co

# plot coefficient progre
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

The dict object creates a dictionary,
and the zip object creates lists.

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.15
 'BLACK': -0.6
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.85
 'DEVIANT1':
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT':
 'GPA1': 0.665
 'HISPANIC':
 'INHEVER1':
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns,

# plot coefficient
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
```

and then in the next set of parenthesis I name the two lists that I want printed.

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.18
 'BLACK': -0.6
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.85
 'DEVIANT1':
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT':
 'GPA1': 0.665
 'HISPANIC':
 'INHEVER1':
 'MALE': -0.21
```

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficients progression
m_log_alphas = -np.log10(model.alpha_)
ax = plt.gca()
```

Then I add a comma and the name of the attribute from the LASSO model results

Out[5]:
{'AGE': 0.2173
'ALCEVR1': -0
'ALCPROBS1':
'ASIAN': 0.18
'BLACK': -0.6
'CIGAVAIL':
'COCEVER1':
'DEP1': -0.85
'DEVIANT1':
'ESTEEM1': 1
'EXPEL1': -0.
'FAMCONCT': 0
'GPA1': 0.665
'HISPANIC': 0
'INHEVER1': 0
'MALE': -0.21

```
predictors['ALCPROBS1']=preprocessing.scale(predictors['ALCPROBS1'].astype('float64'))
predictors['MAREVER1']=preprocessing.scale(predictors['MAREVER1'].astype('float64'))
predictors['COCEVER1']=preprocessing.scale(predictors['COCEVER1'].astype('float64'))
predictors['INHEVER1']=preprocessing.scale(predictors['INHEVER1'].astype('float64'))
predictors['CIGAVAIL']=preprocessing.scale(predictors['CIGAVAIL'].astype('float64'))
predictors['DEP1']=preprocessing.scale(predictors['DEP1'].astype('float64'))
predictors['ESTEEM1']=preprocessing.scale(predictors['ESTEEM1'].astype('float64'))
predictors['VIOL1']=preprocessing.scale(predictors['VIOL1'].astype('float64'))
predictors['PASSIST']=preprocessing.scale(predictors['PASSIST'].astype('float64'))
predictors['DEVIANT1']=preprocessing.scale(predictors['DEVIANT1'].astype('float64'))
predictors['GPA1']=preprocessing.scale(predictors['GPA1'].astype('float64'))
predictors['EXPEL1']=preprocessing.scale(predictors['EXPEL1'].astype('float64'))
predictors['FAMCONCT']=preprocessing.scale(predictors['FAMCONCT'].astype('float64'))
predictors['PARACTV']=preprocessing.scale(predictors['PARACTV'].astype('float64'))
predictors['PARPRES']=preprocessing.scale(predictors['PARPRES'].astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alpha
ax = plt.gca()
```

Which is the name of the regression co-efficient attribute.

Object inspecto

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173
 'ALCEVR1': -0
 'ALCPROBS1':
 'ASIAN': 0.18
 'BLACK': -0.6
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.85
 'DEVIANT1':
 'ESTEEM1': 1.
 'EXPEL1': -0.
 'FAMCONCT': 0
 'GPA1': 0.665
 'HISPANIC': 0
 'INHEVER1': 0
 'MALE': -0.21
```

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0.0}
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 68 Column: 37 Memory: 32 %

Predictors with regression coefficients equal to zero

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0.0}
```

means that the coefficients for

Line: 68 Column: 37 Memory: 32 %

those variables had shrunk to zero after

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926710022800039,
```

applying the LASSO regression penalty, and were subsequently removed from the model.

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926713022800639,
 'WHITE': 0.0}
```

So the results show that
of the 23 variables,

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.1098387918895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0.0}
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 68 Column: 37 Memory: 32 %

18 were selected in the final model.

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0.
```

we standardized the values of our variables to be on the same scale.

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.1098387918895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0
```

So we can also use the size of the regression coefficients to tell us which

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.1098387918895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800630,
 'WHITE': 0.0}
```

predictors are the strongest predictors of school connectedness.

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.10983879188895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
```

For example, self-esteem and depression had the largest regression coefficients,

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.1098387918895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.2030047450001038 5,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE':
```

and were most strongly associated

with school connectedness,

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.2173410206527539,
 'ALCEVR1': -0.32499649609663234,
 'ALCPROBS1': 0.0,
 'ASIAN': 0.18869030694621877,
 'BLACK': -0.69364786863865402,
 'CIGAVAIL': -0.1098387918895129,
 'COCEVER1': -0.20000921703106631,
 'DEP1': -0.85417844475685412,
 'DEVIANT1': -0.41808246027790863,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.07382899854862173 8,
 'FAMCONCT': 0.51527294787432376,
 'GPA1': 0.66557641766975117,
 'HISPANIC': 0.20300474500010385,
 'INHEVER1': 0.0,
 'MALE': -0.21508693783564239,
 'MAREVER1': -0.15980377488872288,
 'NAMERICAN': -0.10784573616427021,
 'PARACTV': 0.29991192982913367,
 'PARPRES': 0.0,
 'PASSIST': 0.0,
 'VIOL1': -0.63926717022800639,
 'WHITE': 0.0}
```

followed by black ethnicity and GPA.

```python
# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, target, test_size=.3,
                                                              random_state=123)

# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')


# RSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)
```

```
Out[5]:
{'AGE': 0.217941020052753 9,
 'ALCEVR1': -0.3240964060066323 4,
 'ALCPROB1': 0.0,
 'ASIAN': 0.1886903069652187 7,
 'BLACK': -0.0956470686380540 3,
 'CIGAVAIL': -0.1098387918895139,
 'COCEVER1': -0.2000062170318663 1,
 'DEP1': -0.0541784647568541 2,
 'DEVIANT1': -0.4188824002779006 3,
 'ESTEEM1': 1.0974098143740421,
 'EXPEL1': -0.0788289854802137 8,
 'FAMCONCT': 0.5152720478743237 6,
 'GPA1': 0.0055764176697511 7,
 'HISPANIC': 0.2020047450001838 5,
 'INHEVER1': 0.0,
 'MALE': -0.2150069378356423 9,
 'MARIJ1': -0.1598097348887218 8,
 'NAMERICAN': -0.1078457361642702 1,
 'PARACTV': 0.2008118245207376 7,
 'VIOL1': -0.6392671702280063 9,
 'WHITE': 0.0}
```

We can also create some plots so we can visualize some of the results.

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

For example, we can plot the progression of the regression coefficients through

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROBS1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

by values of the penalty parameter at each step of the selection process.

Object inspector

Python console

Console 1/A

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROBS1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

It's important to note that the sklearn library refers to the penalty

Object inspector

Python console

Console 1/A

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROBS1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(al
plt.ylabel('Mean squared error')
plt.title('Mean squared err
```

Sklearn: penalty parameter is called ***alpha***

Object inspector

IPython console

Console 1/A

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROBS1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

parameter is alpha although the more conventional term is lambda.

```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

We can use the following code to generate this plot.

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROB1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

is equal to the negative of
the -np.log10 transformation

Out[5]:
{'AGE': 0.21
'ALCEVR1':
'ALCPROB1'
'ASIAN': 0.
'BLACK': -0
'CIGAVAIL':
'COCEVER1':
'DEP1': -0.
'DEVIANT1':
'ESTEEM1':
'EXPEL1':
'FAMCONCT':

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean s
plt.title('Mean squared error on each fold')
```

function applied to the alphas_
attribute from the model results object.

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROB1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

The alphas_ attribute

Out[5]:
{'AGE': 0.21
'ALCEVR1':
'ALCPROB1'
'ASIAN': 0.
'BLACK': -0
'CIGAVAIL':
'COCEVER1':
'DEP1': -0.
'DEVIANT1':
'ESTEEM1':
'EXPEL1':
'FAMCONCT':

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

contains the values of alpha through the model selection process.

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean Squared Error')
plt.title('Mean Squared Error on each fold')
```

The first line of code for the plot, sets up the axes, the second line of code asks

```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean
plt.title('Mean squared error on each fold')
```

Python to use the plot function from the mat plot lib library which we imported as

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

plt to plot the transform values of alpha on the horizontal access.

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROB1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

And the change in the regression coefficients in the coef_path_ attribute,

```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

The .T asks python to transpose the coef_path_ attribute

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

matrix to match the first dimension
of the array of alpha values.

```
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

I will use the plt.axlvline function to put a dashed vertical line at

```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

the -np.log10 transformed alpha value for the selected model.

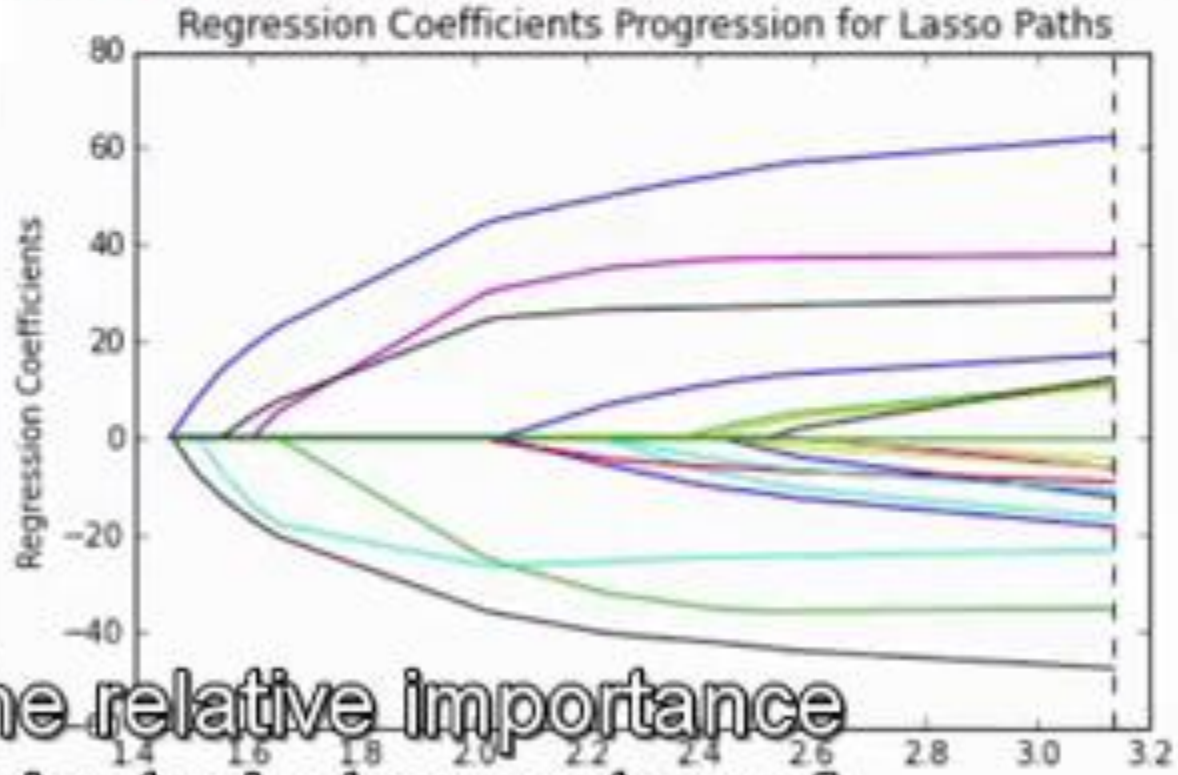
```python
# specify the lasso regression model
model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

# print variable names and regression coefficients
dict(zip(predictors.columns, model.coef_))

# plot coefficient progression
m_log_alphas = -np.log10(model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, model.coef_path_.T)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squa
plt.title('Mean squared error on each fold')
```

The color equals='k' in quotes tells Python to make the line color black.

```
Out[5]:
{'AGE': 0.21
 'ALCEVR1':
 'ALCPROB1'
 'ASIAN': 0.
 'BLACK': -0
 'CIGAVAIL':
 'COCEVER1':
 'DEP1': -0.
 'DEVIANT1':
 'ESTEEM1':
 'EXPEL1': -
 'FAMCONCT':
```

```
, 'k',

or='k',

(pred_train))
pred_test))
```

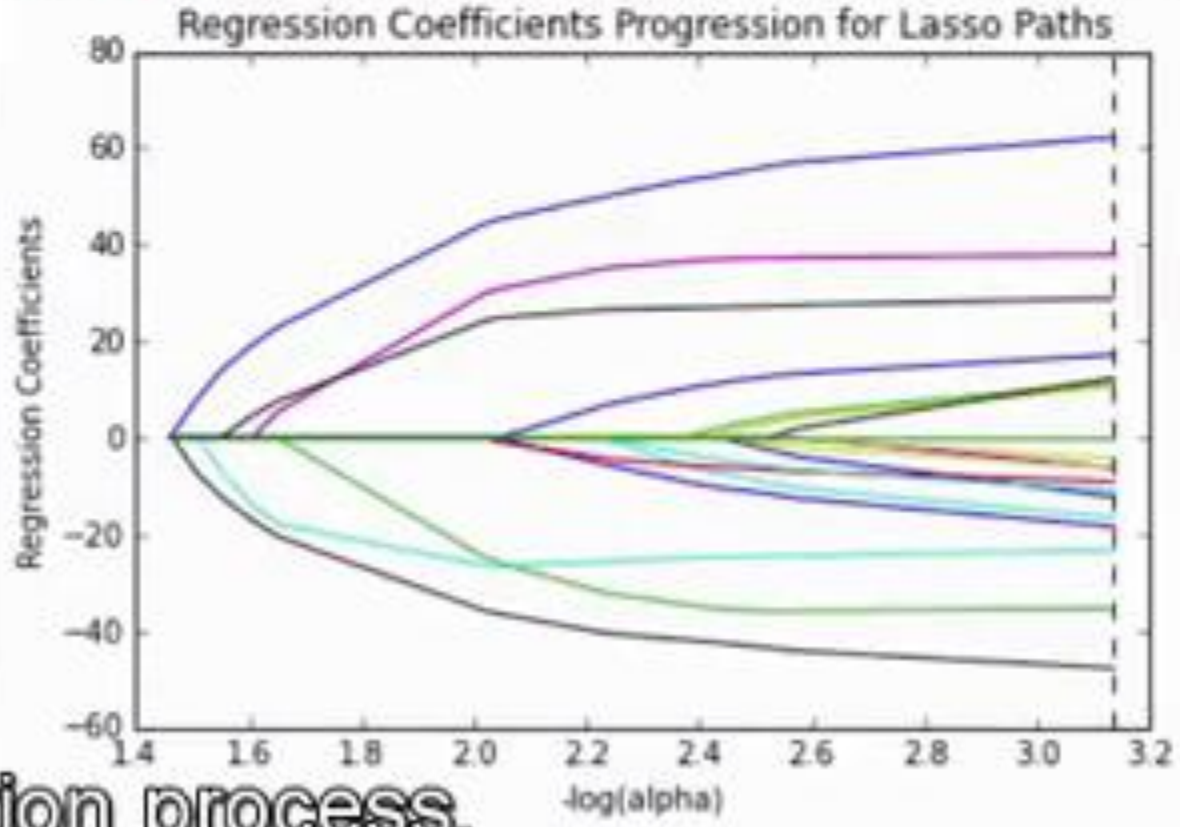
```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



This plot shows the relative importance of the predictor selected at any step of

```
, 'k',

or='k',

(pred_train))
pred_test))
```

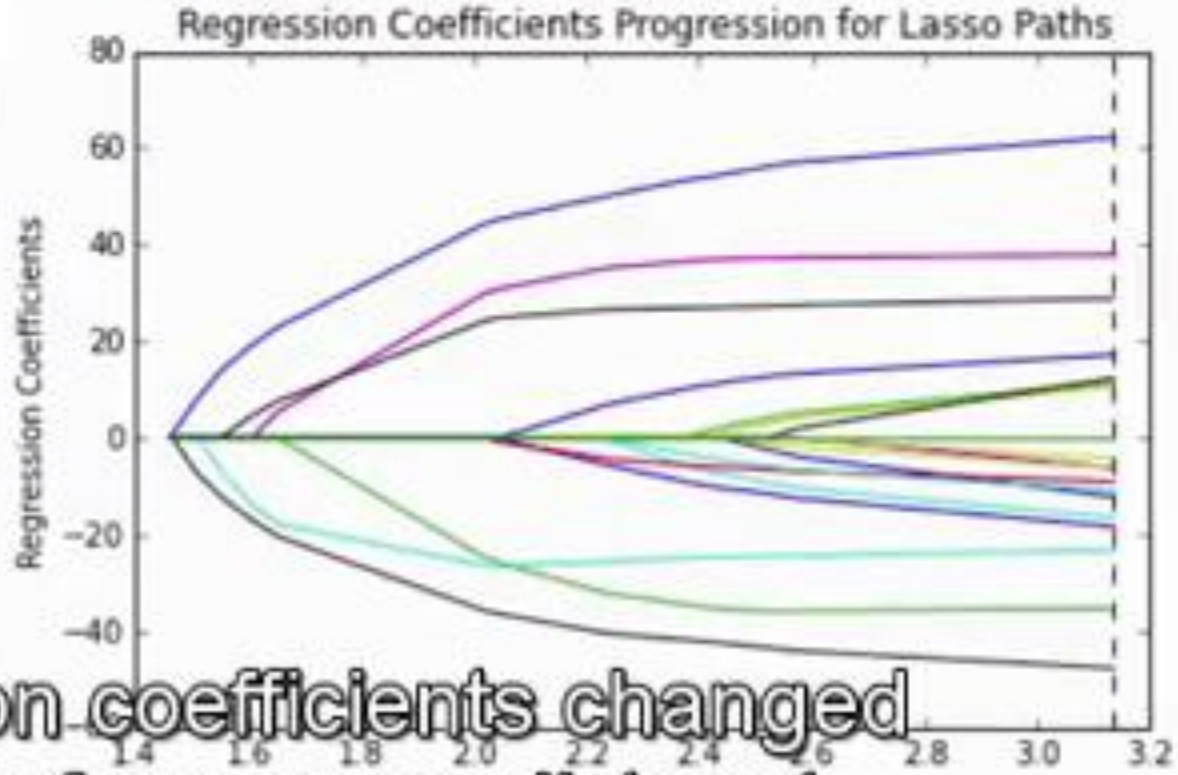
```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



the selection process,

```
, 'k',

or='k',

(pred_train))
pred_test))
```

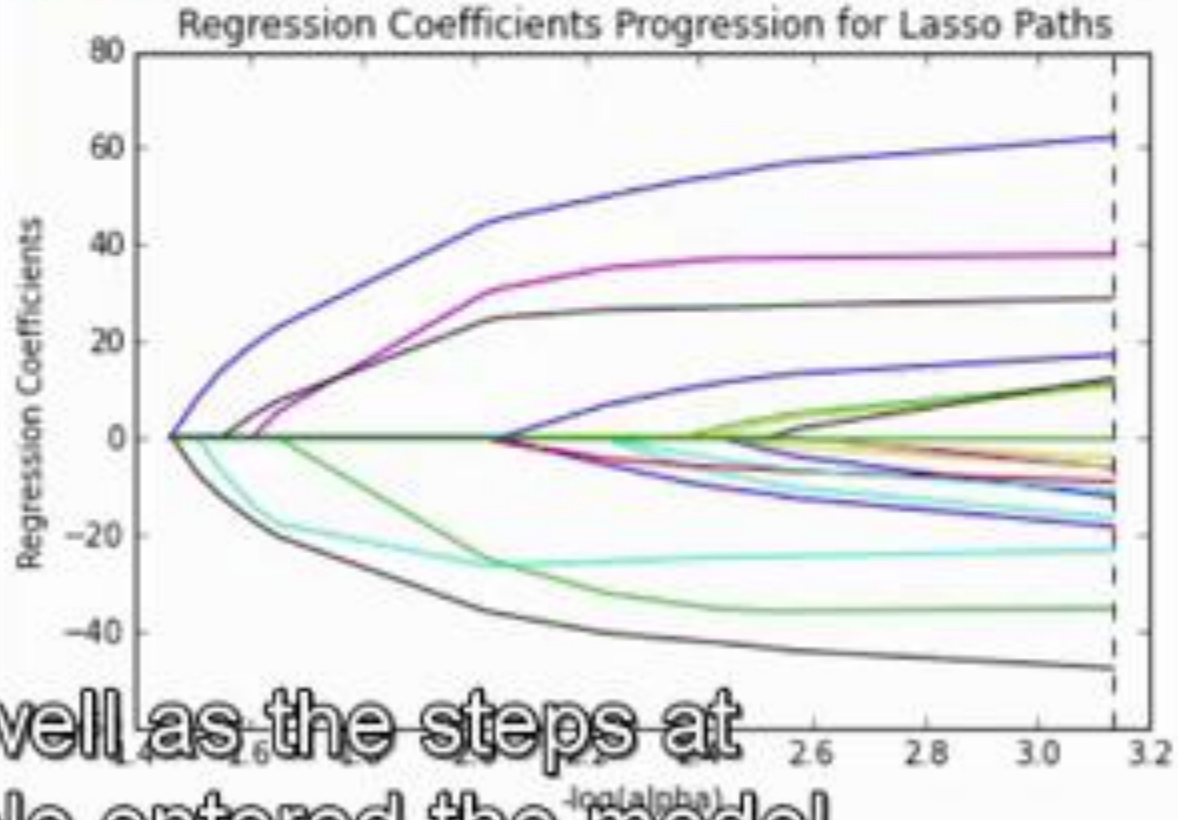
```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



how the reggression coefficients changed
with the addition of a new predictor at

```
, 'k',

or='k',

(pred_train))
pred_test))
```

```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```

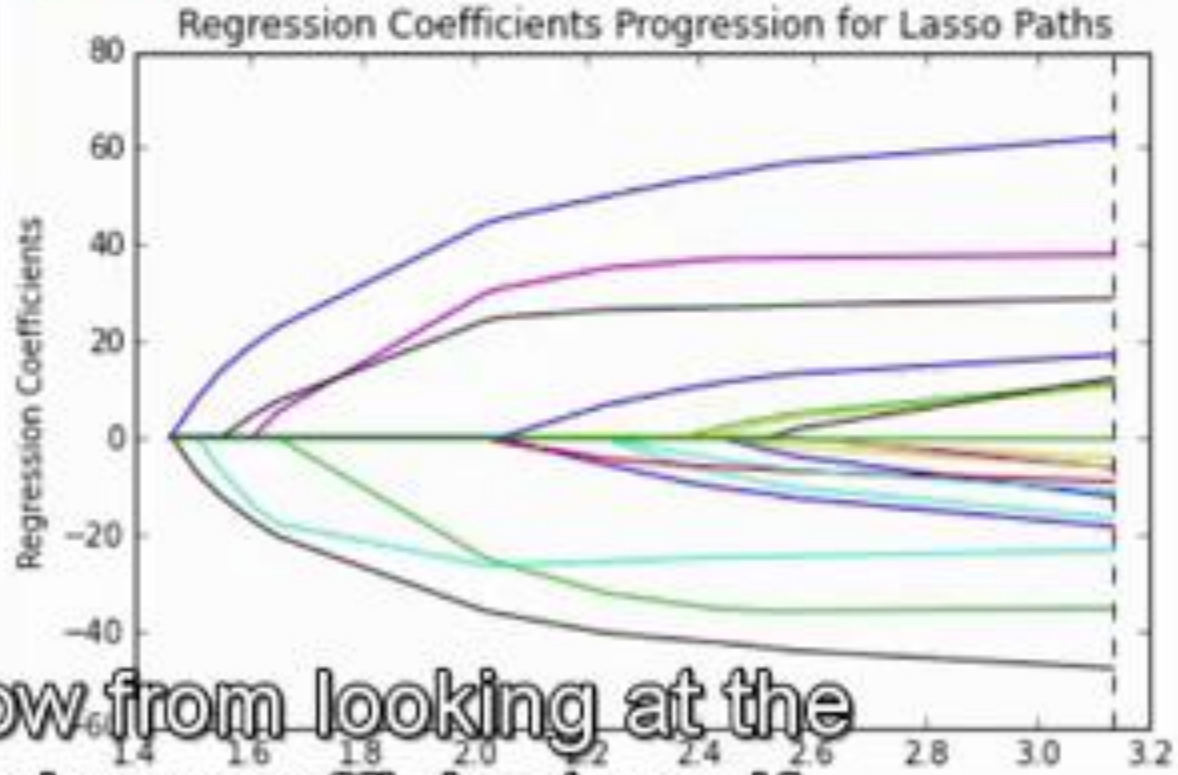


each step, as well as the steps at which each variable entered the model.

In [7]:

```
, 'k',

or='k',

(pred_train))
pred_test))
```

```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```

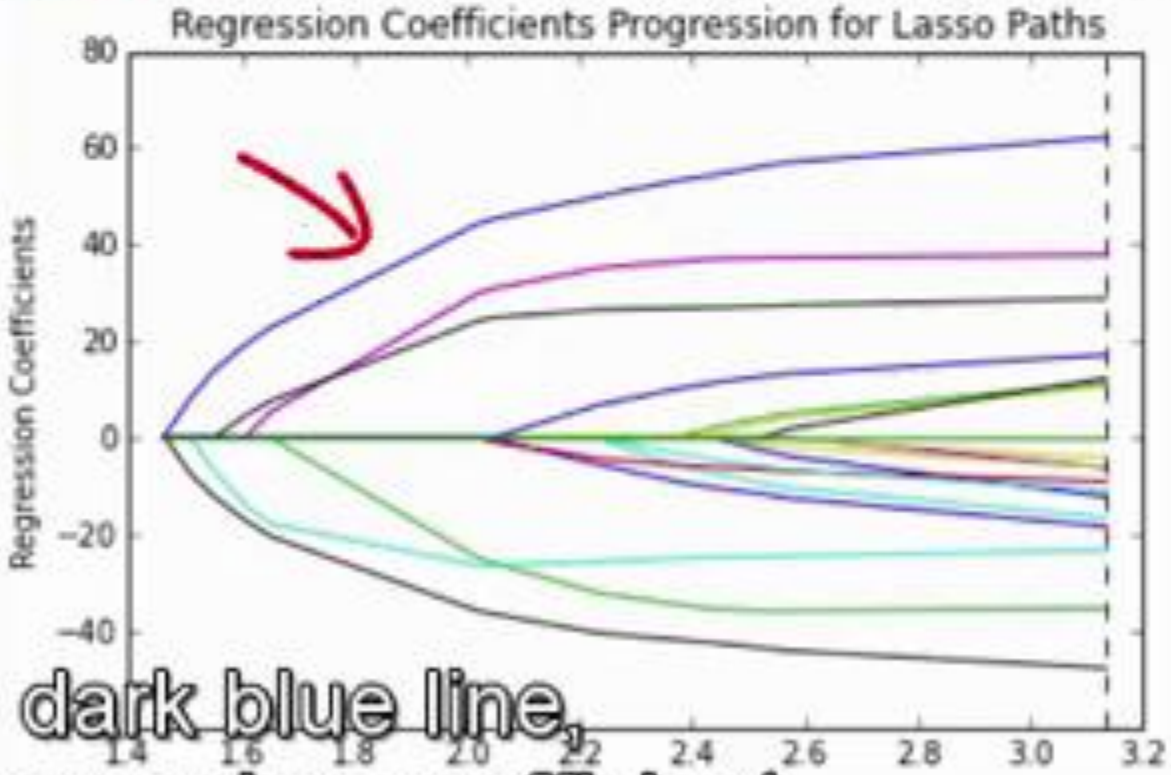


As we already know from looking at the list of the regression coefficients self

In [7]:

```
, 'k',

or='k',

(pred_train))
pred_test))
```

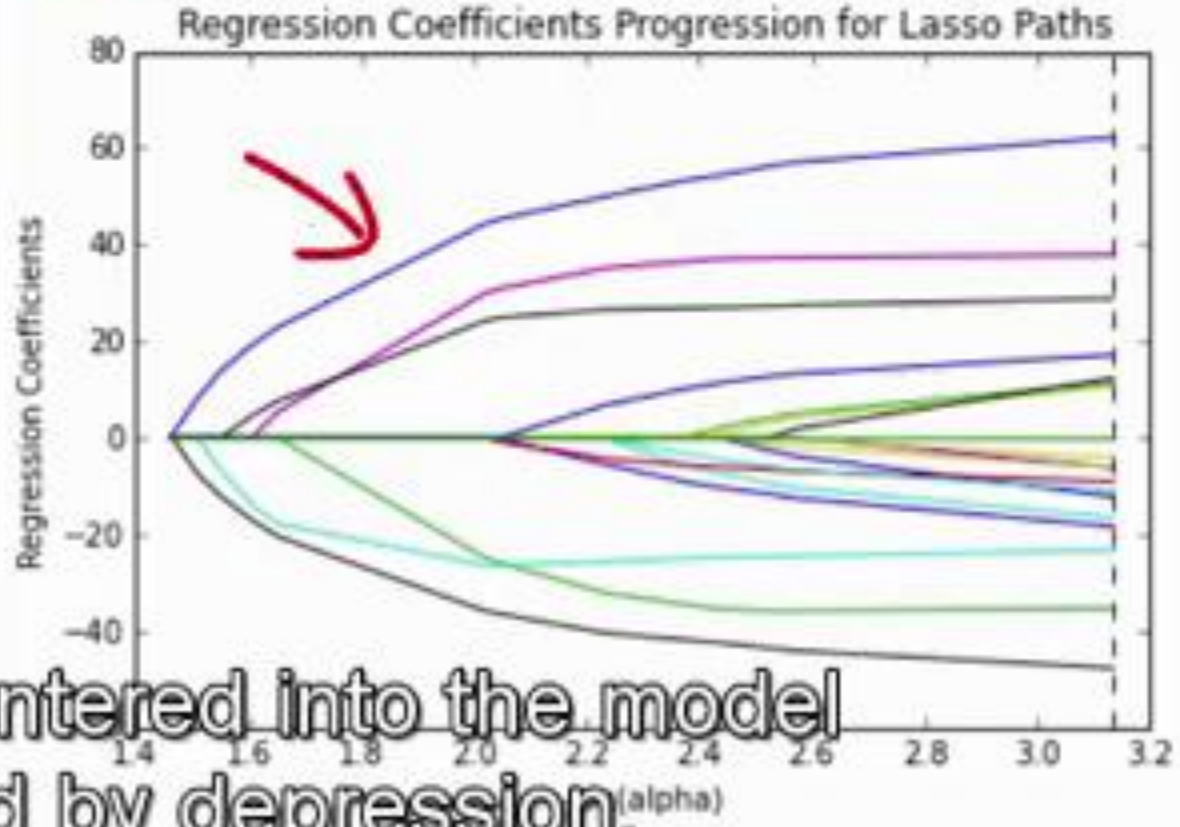
```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



esteem, the dark blue line,
had the largest regression coefficient.

```
, 'k',

or='k',

(pred_train))
pred_test))
```

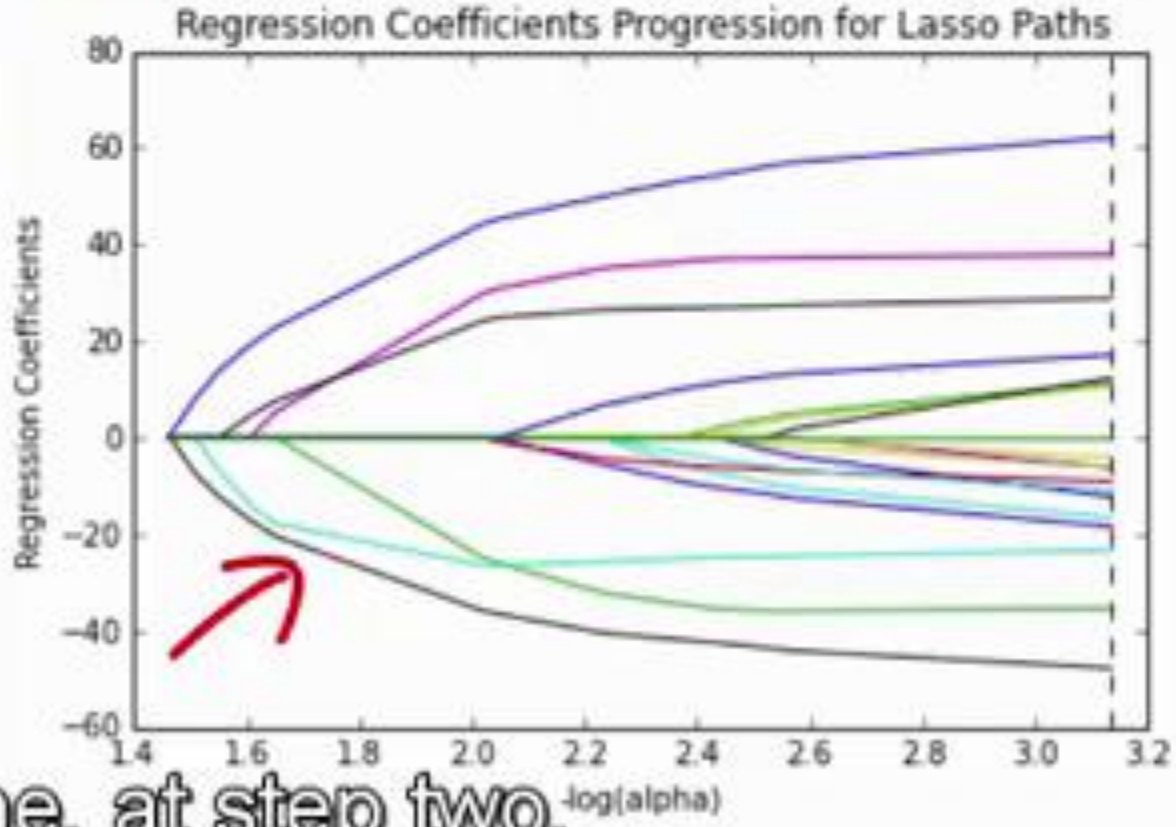
```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



It was therefore entered into the model first, followed by depression,

```
, 'k',

or='k',

(pred_train))
pred_test))
```

```
...: plt.plot(m_log_alphas, model.coef_path_.T)
...: plt.axvline(-np.log10(model.alpha_), linest
...:             label='alpha CV')
...: plt.ylabel('Regression Coefficients')
...: plt.xlabel('-log(alpha)')
...: plt.title('Regression Coefficients Progress
...:
Out[6]: <matplotlib.text.Text at 0xd0c3780>
```



the black line, at step two.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')
```

```
# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

```
# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(pred_train))
test_error = mean_squared_error(y_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

Another important plot is one that shows the change in the mean square error for

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(scaled_train))
test_error = mean_squared_error(y_test, model.predict(scaled_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

the change in the penalty parameter alpha
at each step in the selection process.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

the previous plot except this time
we're plotting the alpha values through

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

the model selection process for each cross validation fold on the horizontal axis,

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = m
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

and the mean square error for each cross validation fold on the vertical axis.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

This is done in the first plt.plot function.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

Where m_log_alphascv,

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

is a negative log10 transformation applied to the alphascv_ attribute for

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

each validation fold, and
a cv_mse_path_ is the model results

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

attribute containing the mean square error for each cross validation fold.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

The colon in quotes here tells Python to plot the folds as dotted lines.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(target_train, model.predict(prod_train))
test_error = mean_squared_error(tar_test, model.predict(prod_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

In the next line of code, I'm asking Python to plot the average mean squared

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(y_train, model.predict(X_train))
test_error = mean_squared_error(y_test, model.predict(X_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

to plot it as a slightly thicker line with equals to black line.

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

I'll use the plt.axvline function
to plot a dashed black vertical

```
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')

# plot mean square error for each fold
m_log_alphascv = -np.log10(model.cv_alphas_)
plt.figure()
plt.plot(m_log_alphascv, model.cv_mse_path_, ':')
plt.plot(m_log_alphascv, model.cv_mse_path_.mean(axis=-1), 'k',
         label='Average across the folds', linewidth=2)
plt.axvline(-np.log10(model.alpha_), linestyle='--', color='k',
            label='alpha CV')
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
```

line at the -np.log10 transformed alpha value for the selected model.

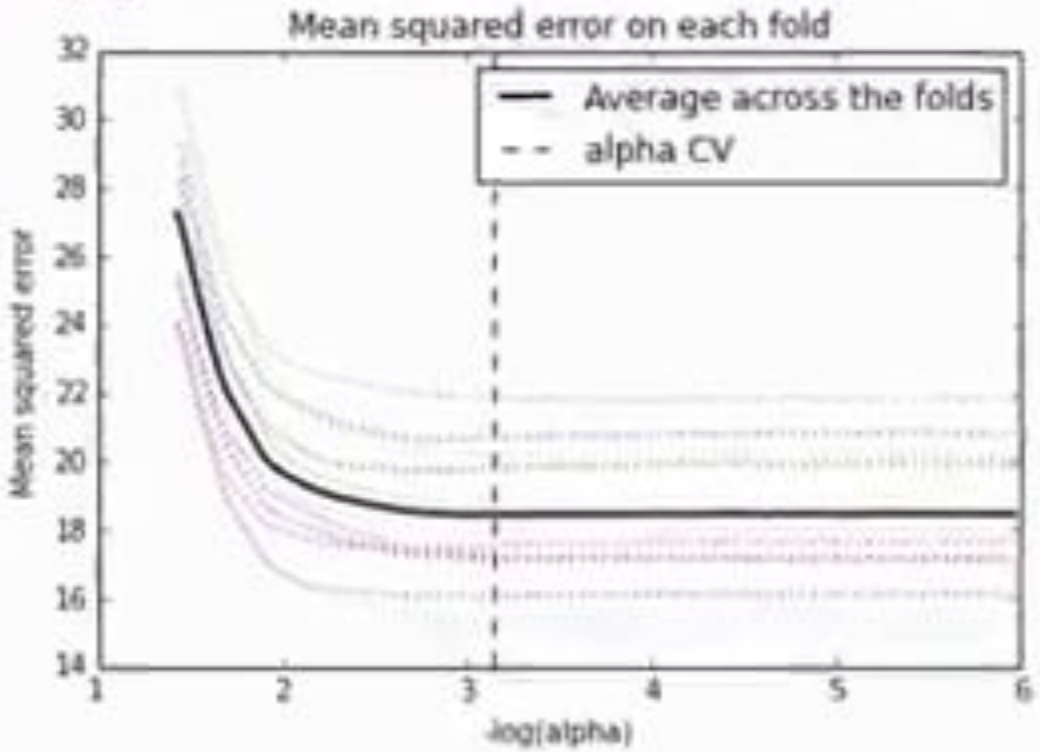
```
k',

k',

red_train))
l_test))
```

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
RuntimeWarning: divide by zero encountered in log10
  # -*- coding: utf-8 -*-
Out[2]: <matplotlib.text.Text at 0xc231080>
```

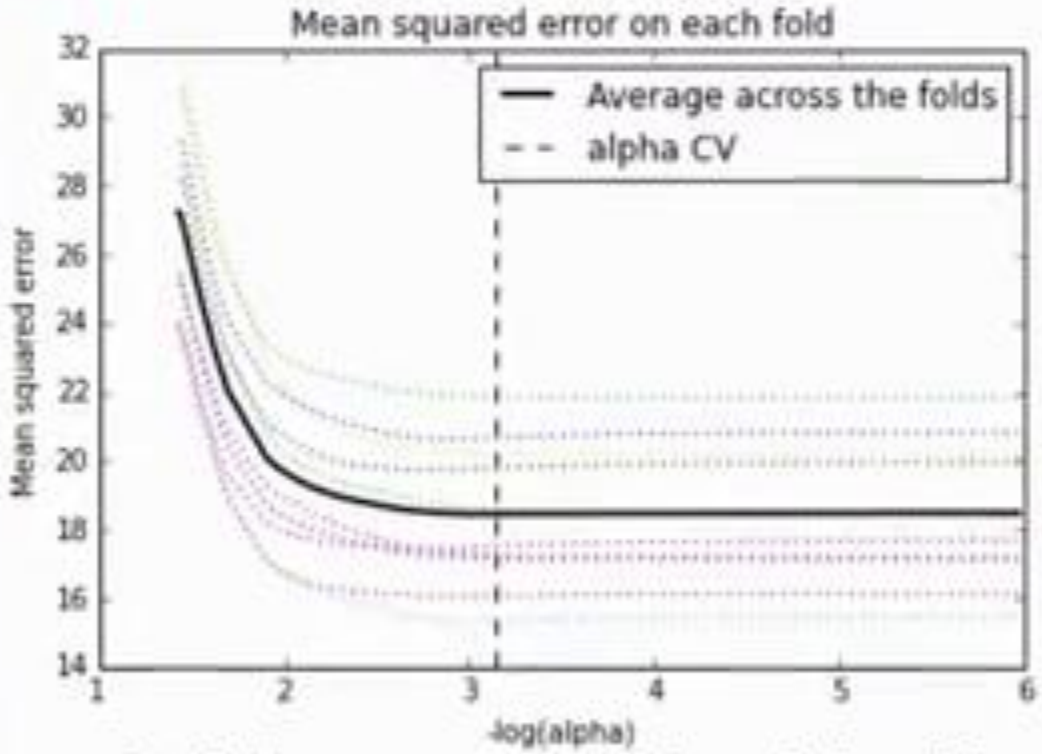


We can see that there is variability across the individual cross-validation

IPython console

Console 1/A

```
C:\Users\jrose01\AppData\Local\Continuum\Anaconda3\lib\site-
packages\spyderlib\widgets\externalshell\start_ipython_kernel.py:1:
RuntimeWarning: divide by zero encountered in log10
  # -*- coding: utf-8 -*-
Out[2]: <matplotlib.text.Text at 0xc231080>
```



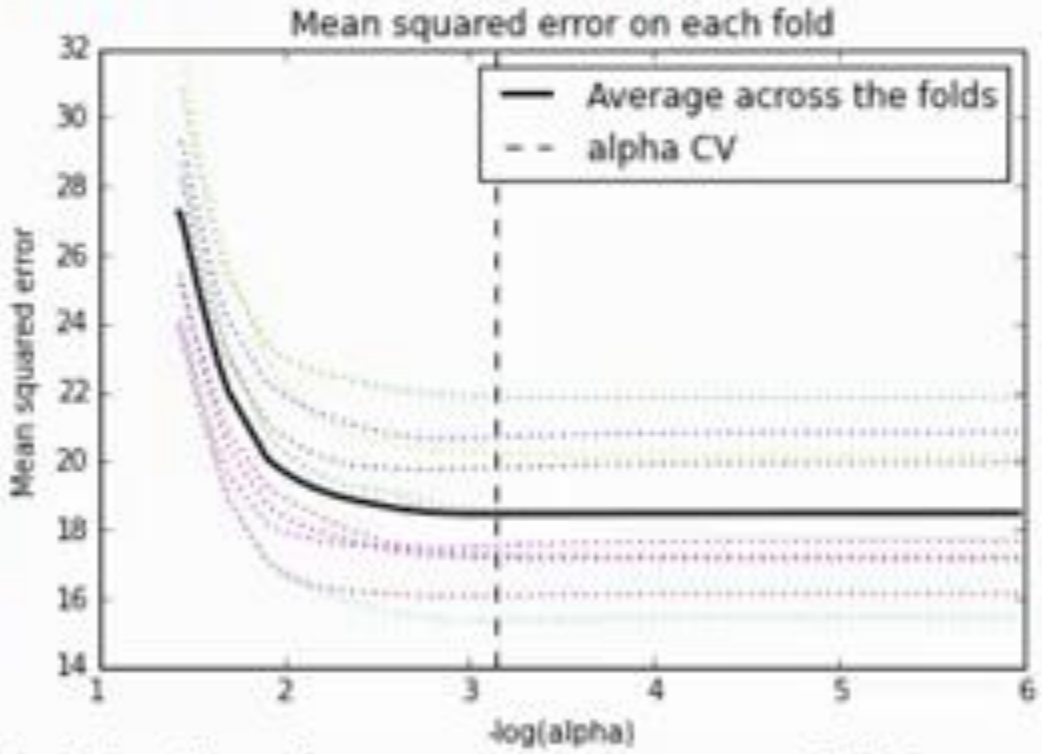
error as variables are added to the model
follows the same pattern for each fold.

In [3]:

IPython console

Console 1/A

```
C:\Users\jrose01\AppData\Local\Continuum\Anaconda3\lib\site-
packages\spyderlib\widgets\externalshell\start_ipython_kernel.py:1:
RuntimeWarning: divide by zero encountered in log10
  # -*- coding: utf-8 -*-
Out[2]: <matplotlib.text.Text at 0xc231080>
```



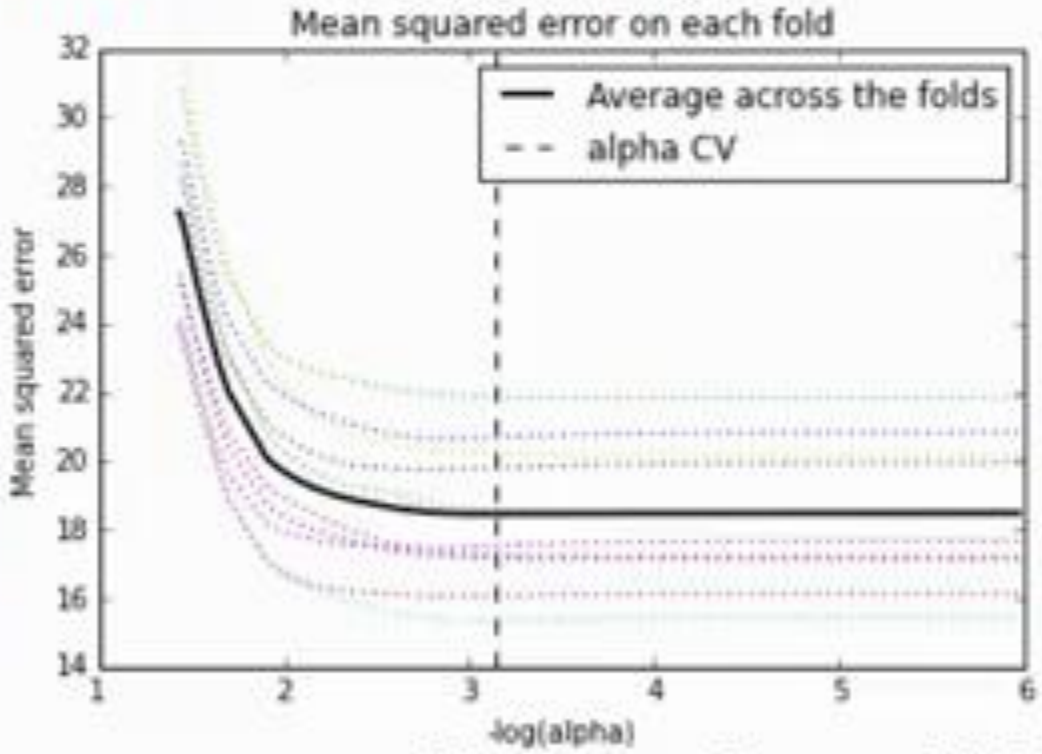
Initially it decreases rapidly and then levels off to a point at which

```
In [3]:
```

IPython console

Console 1/A

```
C:\Users\jrose01\AppData\Local\Continuum\Anaconda3\lib\site-
packages\spyderlib\widgets\externalshell\start_ipython_kernel.py:1:
RuntimeWarning: divide by zero encountered in log10
  # -*- coding: utf-8 -*-
Out[2]: <matplotlib.text.Text at 0xc231080>
```



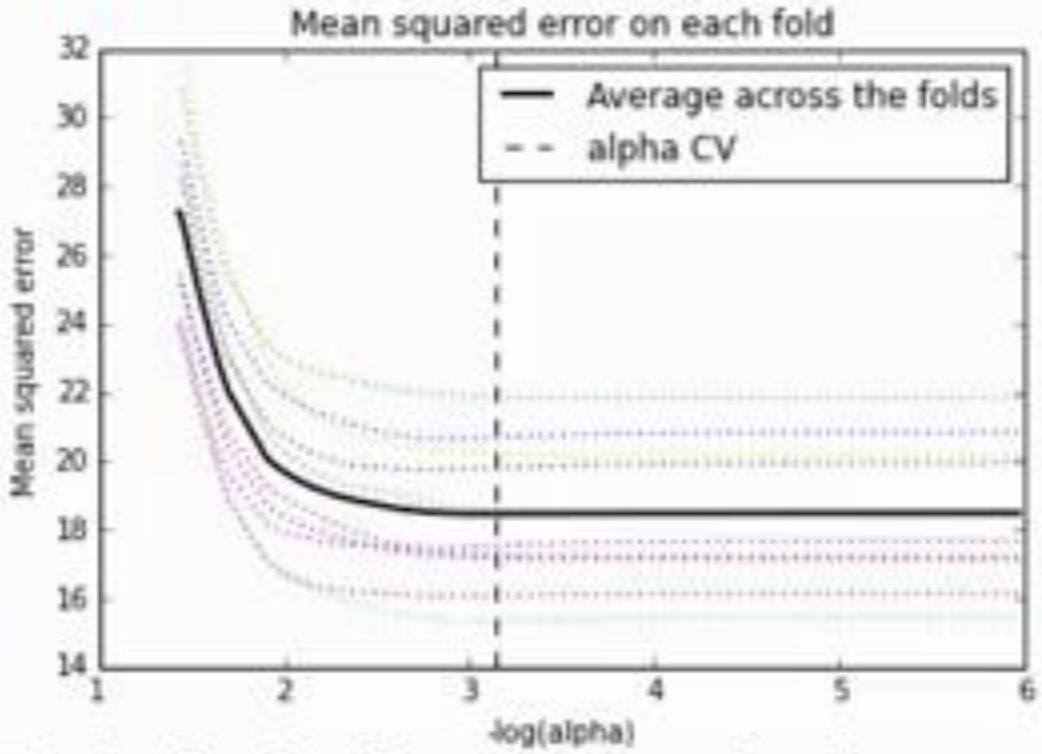

```
In [3]:
```

adding more predictors doesn't lead to much reduction in the mean square error.

IPython console

Console 1/A

```
C:\Users\jrose01\AppData\Local\Continuum\Anaconda3\lib\site-
packages\spyderlib\widgets\externalshell\start_ipython_kernel.py:1:
RuntimeWarning: divide by zero encountered in log10
  # -*- coding: utf-8 -*-
Out[2]: <matplotlib.text.Text at 0xc231080>
```



This is to be expected as model complexity increases.

In [3]:

```
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

```
# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)

# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

We can also print the average mean square error in the r square for

```
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')
```

```
# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)

# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

That is explained by the selected model
in the training data set and in the test

```
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)

# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

calculation function, then in parentheses,
the training data set response variable,

```python
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)

# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

This code tells Python to use the results from the model object

```
plt.legend()
plt.xlabel('-log(alpha)')
plt.ylabel('Mean squared error')
plt.title('Mean squared error on each fold')

# MSE from training and test data
from sklearn.metrics import mean_squared_error
train_error = mean_squared_error(tar_train, model.predict(pred_train))
test_error = mean_squared_error(tar_test, model.predict(pred_test))
print ('training data MSE')
print(train_error)
print ('test data MSE')
print(test_error)

# R-square from training and test data
rsquared_train=model.score(pred_train,tar_train)
rsquared_test=model.score(pred_test,tar_test)
print ('training data R-square')
print(rsquared_train)
print ('test data R-square')
print(rsquared_test)
```

We then do the same thing for the test data by using the results from

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
...: train_error = mean_squared_error(tar_train, model.predict(pred_train))
...: test_error = mean_squared_error(tar_test, model.predict(pred_test))
...: print ('training data MSE')
...: print(train_error)
...: print ('test data MSE')
...: print(test_error)
...:
...: # R-square from training and test data
...: rsquared_train=model.score(pred_train,tar_train)
...: rsquared_test=model.score(pred_test,tar_test)
...: print ('training data R-square')
...: print(rsquared_train)
...: print ('test data R-square')
...: print(rsquared_test)
training data MSE
18.1485726641
test data MSE
17.2925174272
training data R-square
0.333611136927
test data R-square
0.31001113416
```

As expected, the selected model was less accurate in predicting school

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
...: test_error = mean_squared_error(tar_test, model.predict(pred_test))
...: print ('training data MSE')
...: print(train_error)
...: print ('test data MSE')
...: print(test_error)
...:
...: # R-square from training and test data
...: rsquared_train=model.score(pred_train,tar_train)
...: rsquared_test=model.score(pred_test,tar_test)
...: print ('training data R-square')
...: print(rsquared_train)
...: print ('test data R-square')
...: print(rsquared_test)
training data MSE
18.1485726641
test data MSE
17.2925174272
training data R-square
0.333611136927
test data R-square
0.31001113416

In [9]:
```

connectedness in the test data, but

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
...: test_error = mean_squared_error(tar_test, model.predict(pred_test))
...: print ('training data MSE')
...: print(train_error)
...: print ('test data MSE')
...: print(test_error)
...:
...: # R-square from training and test data
...: rsquared_train=model.score(pred_train,tar_train)
...: rsquared_test=model.score(pred_test,tar_test)
...: print ('training data R-square')
...: print(rsquared_train)
...: print ('test data R-square')
...: print(rsquared_test)
training data MSE
18.1485726641
test data MSE
17.2925174272
training data R-square
0.333611136927
test data R-square
0.31001113416

In [9]:
```

This suggests that prediction accuracy was pretty stable across the two data sets.

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
...: test_error = mean_squared_error(tar_test, model.predict(pred_test))
...: print ('training data MSE')
...: print(train_error)
...: print ('test data MSE')
...: print(test_error)
...:
...: # R-square from training and test data
...: rsquared_train=model.score(pred_train,tar_train)
...: rsquared_test=model.score(pred_test,tar_test)
...: print ('training data R-square')
...: print(rsquared_train)
...: print ('test data R-square')
...: print(rsquared_test)
training data MSE
18.1485726641
test data MSE
17.[illegible]
training data R-square
0.333611136927
test data R-square
0.31001113416

In [9]:
```

indicating that the selected model explained 33 and 31% of the variance

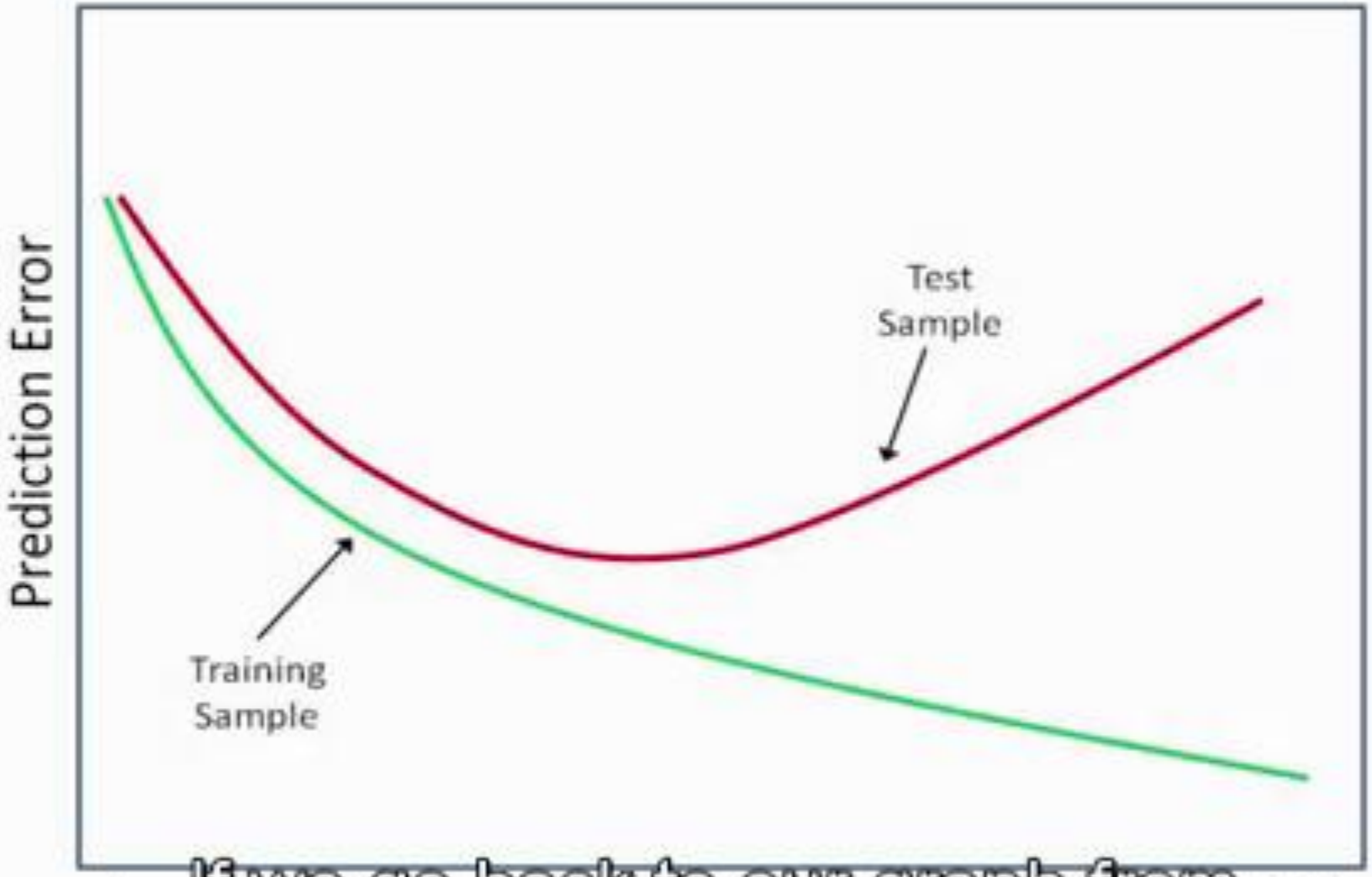
Object inspector | Variable explorer | File explorer

IPython console

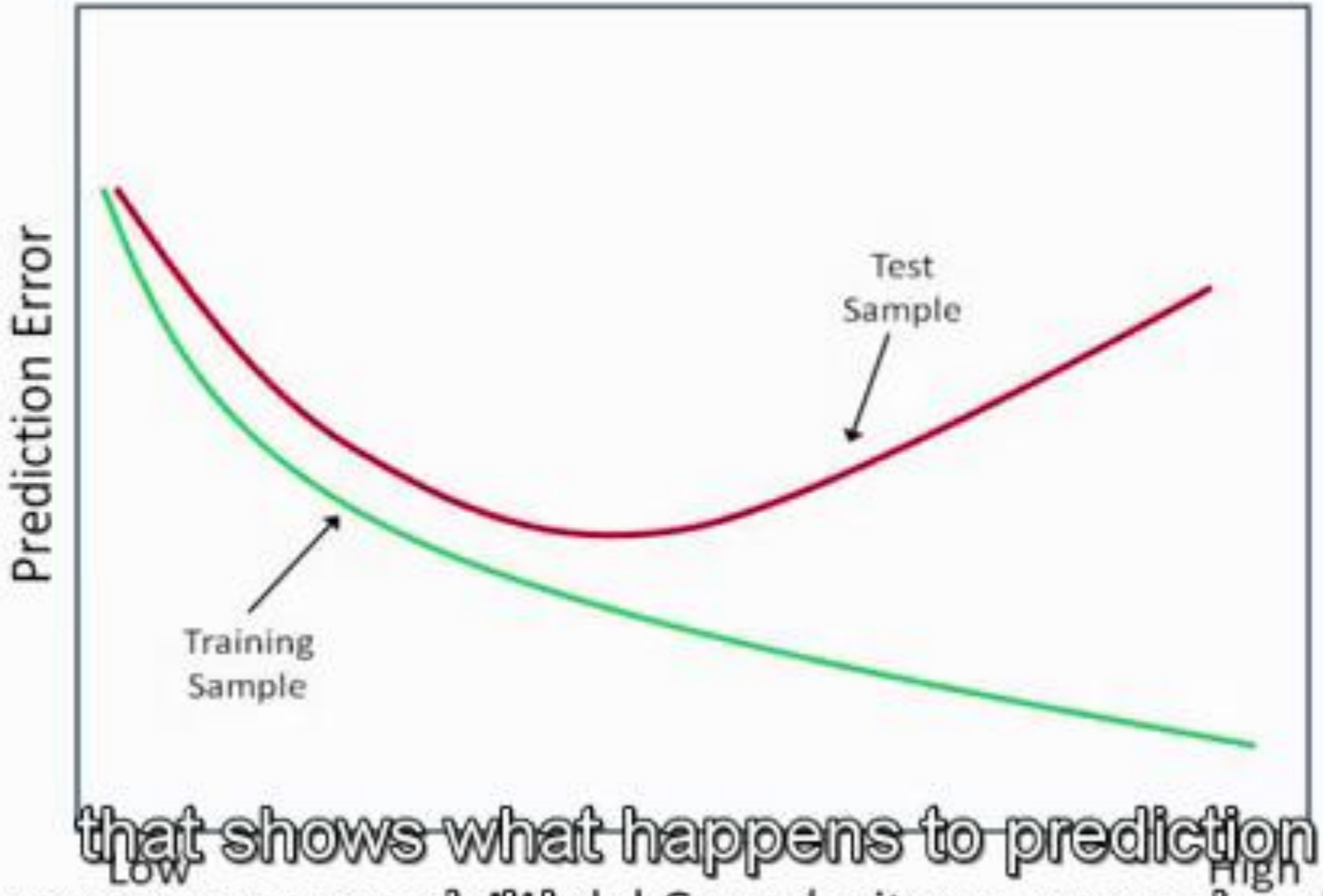
Console 1/A

```
...: test_error = mean_squared_error(tar_test, model.predict(pred_test))
...: print ('training data MSE')
...: print(train_error)
...: print ('test data MSE')
...: print(test_error)
...:
...: # R-square from training and test data
...: rsquared_train=model.score(pred_train,tar_train)
...: rsquared_test=model.score(pred_test,tar_test)
...: print ('training data R-square')
...: print(rsquared_train)
...: print ('test data R-square')
...: print(rsquared_test)
training data MSE
18.1485726641
test data MSE
17.[illegible]
training data R-square
0.333611136927
test data R-square
0.31001113416

In [9]:
```

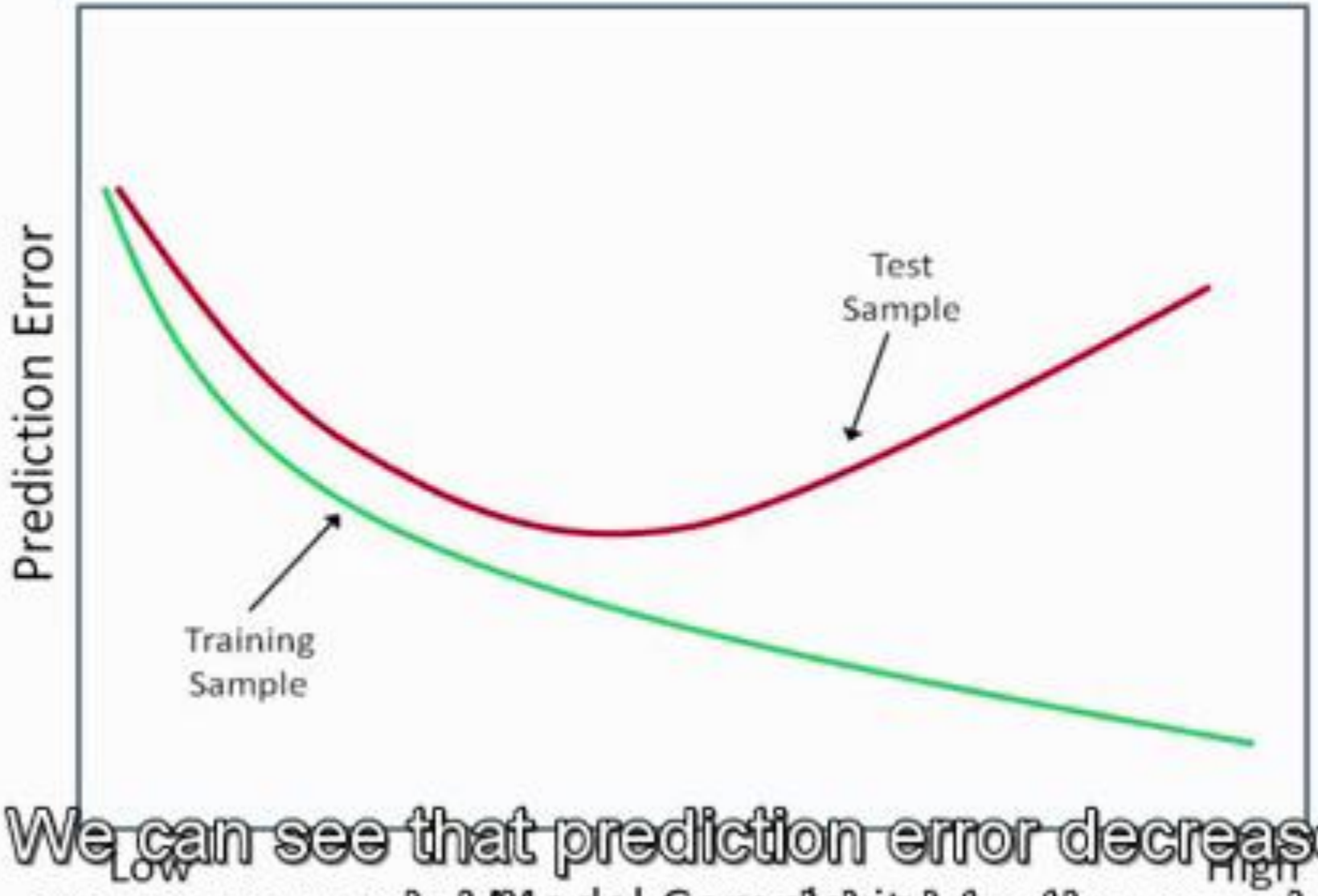
in school connectedness for the training and test sets, respectively.



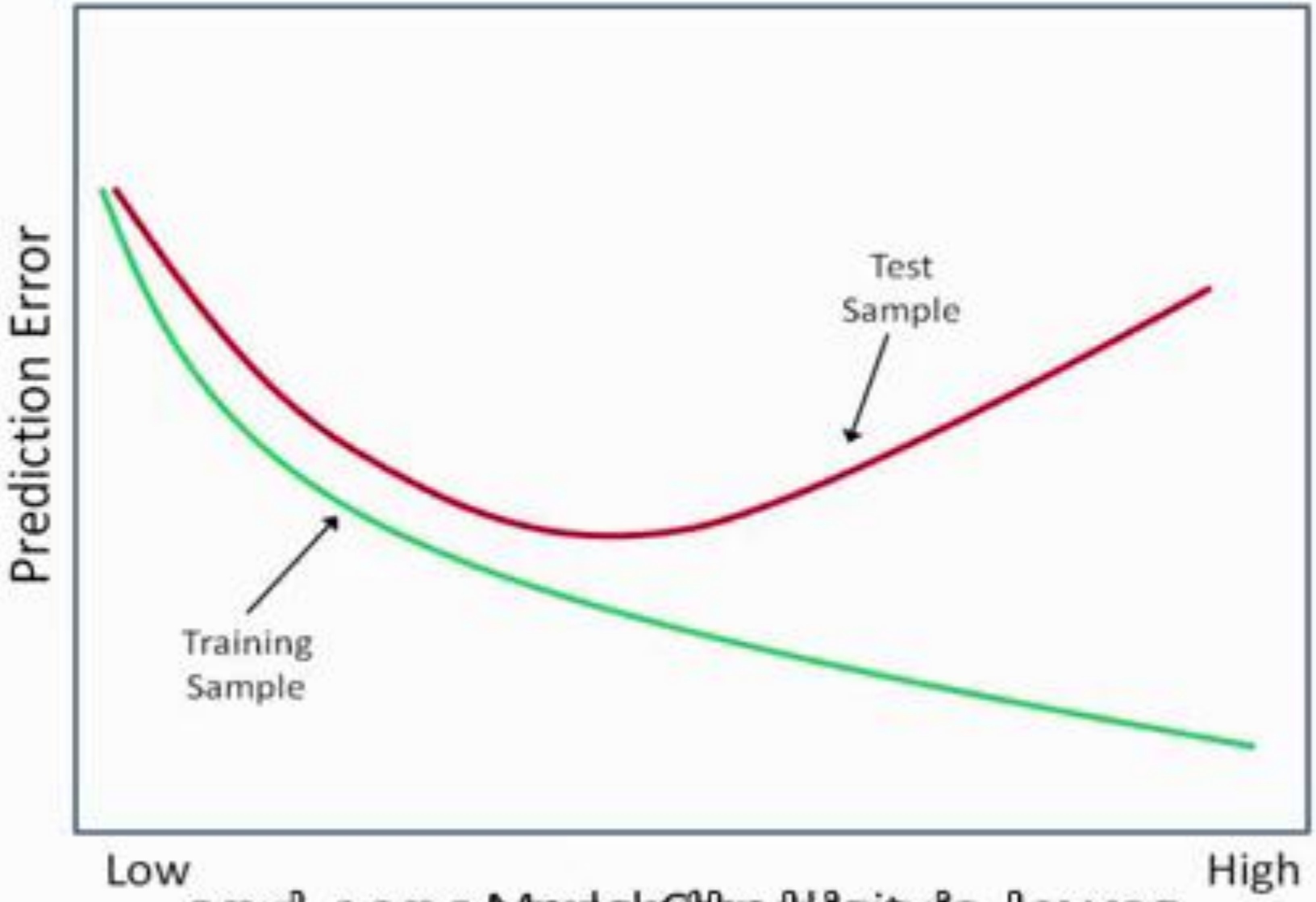
If we go back to our graph from the bias variance trade-off video



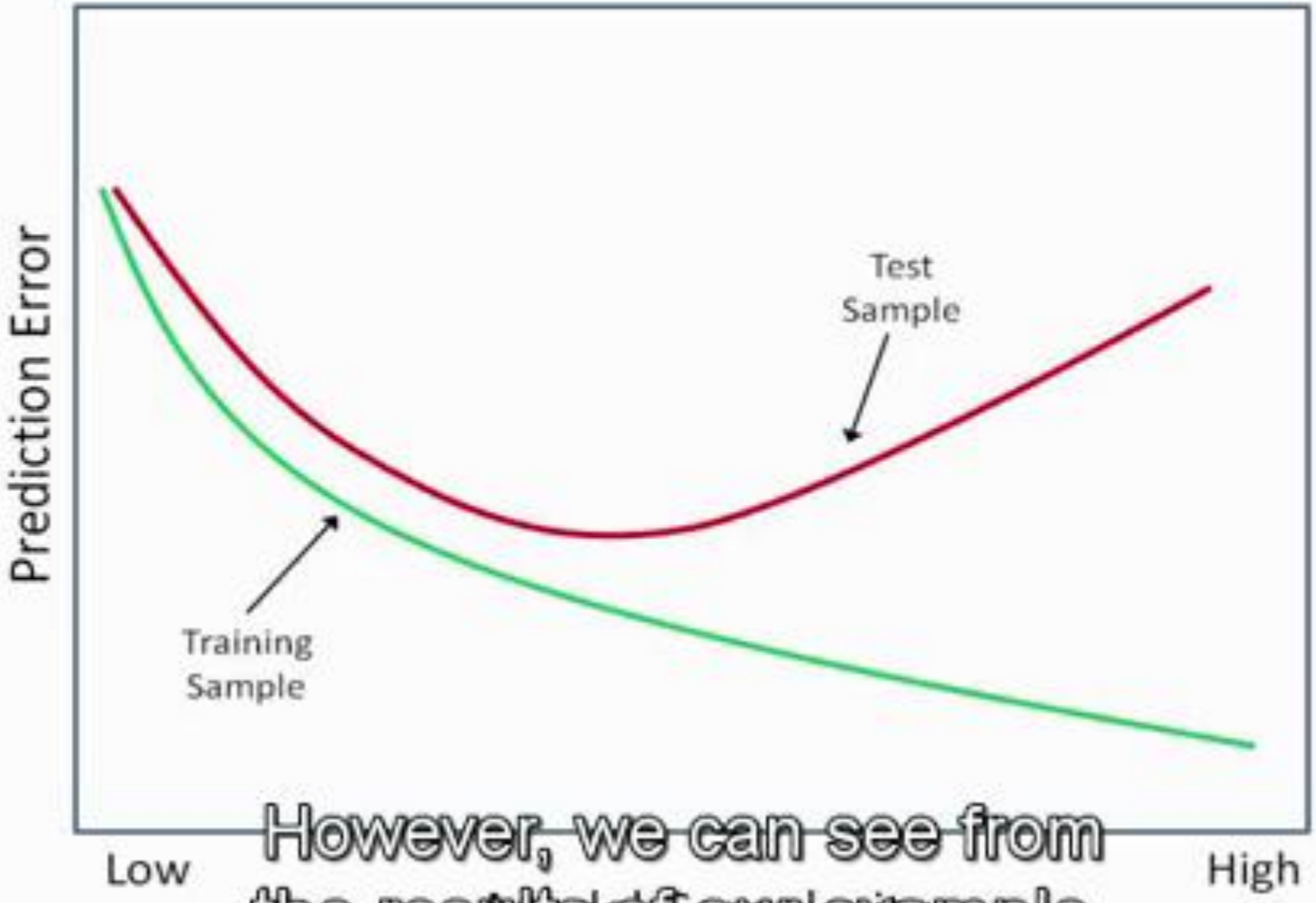
that shows what happens to prediction error as a model becomes more complex by



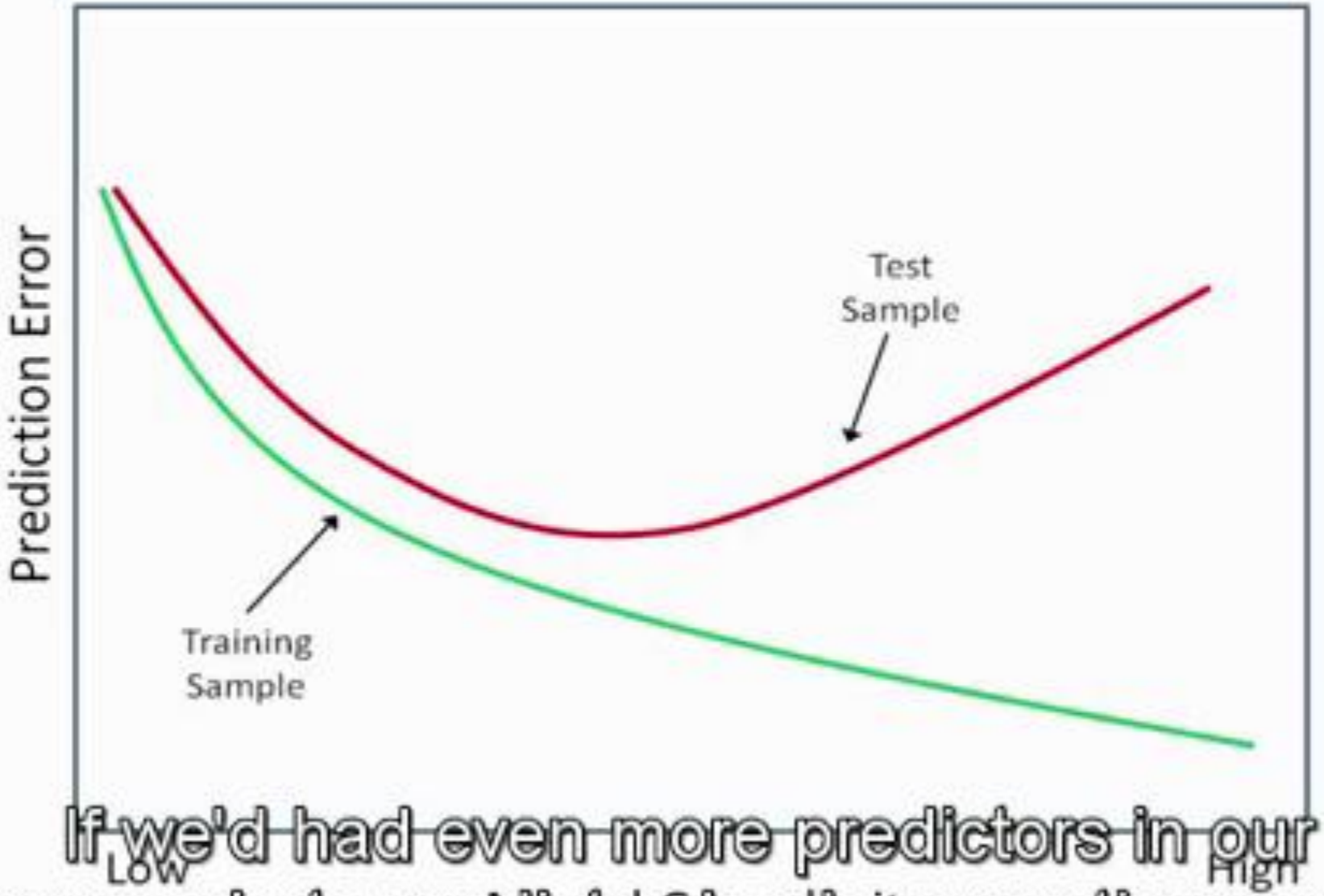
We can see that prediction error decreases as more variables are added to the model,



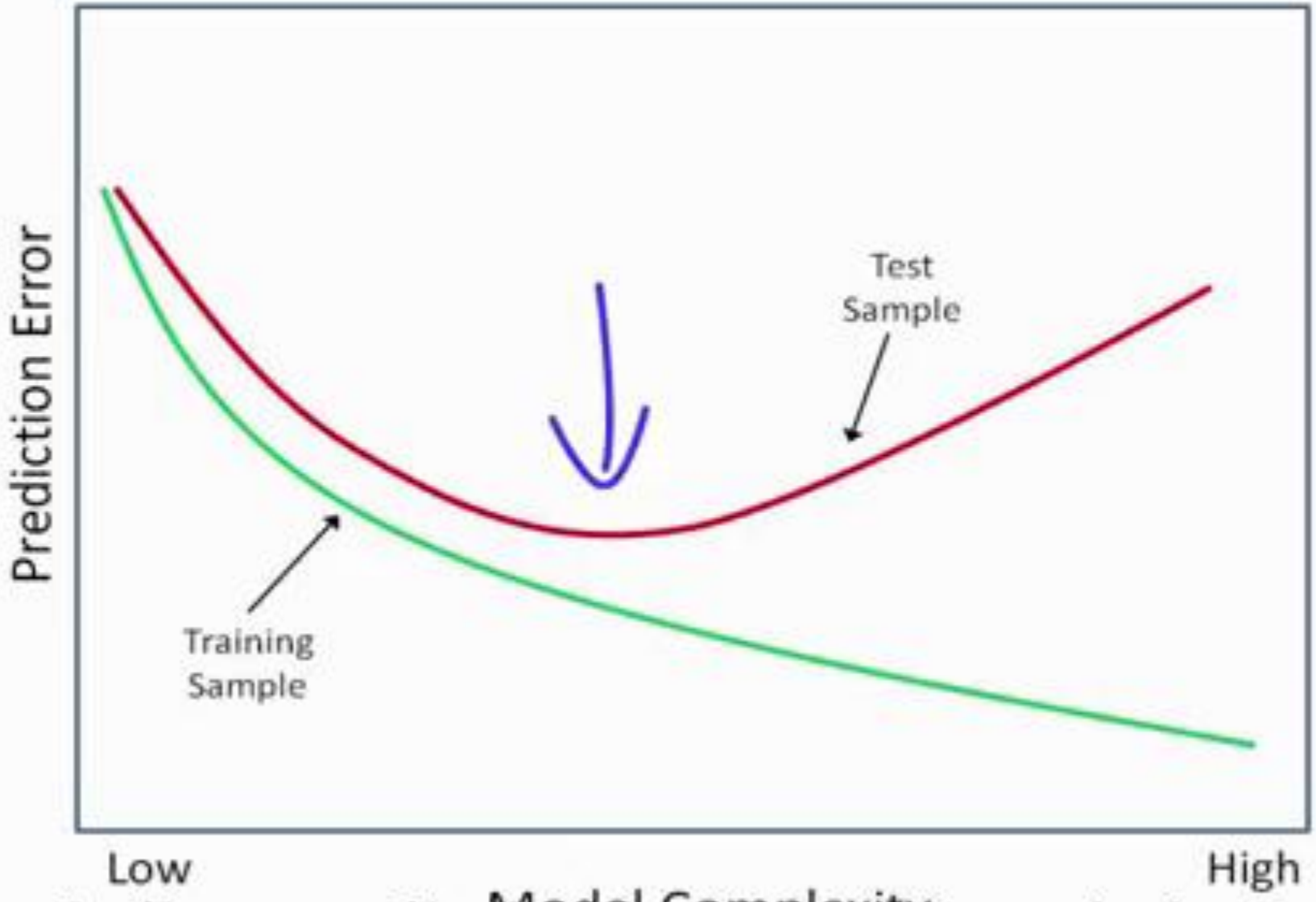
and consequently bias is lower.



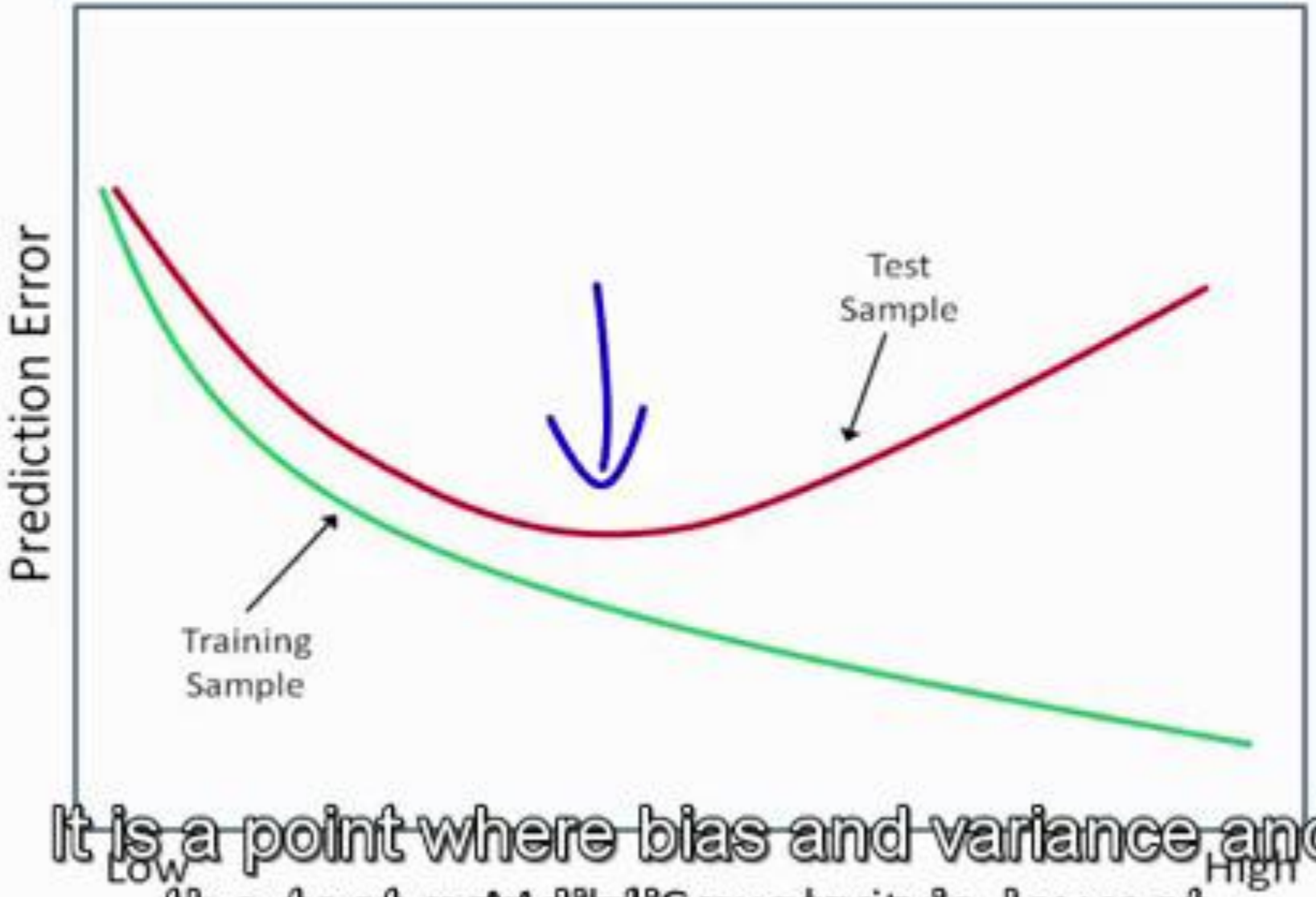
However, we can see from
the results of our example,



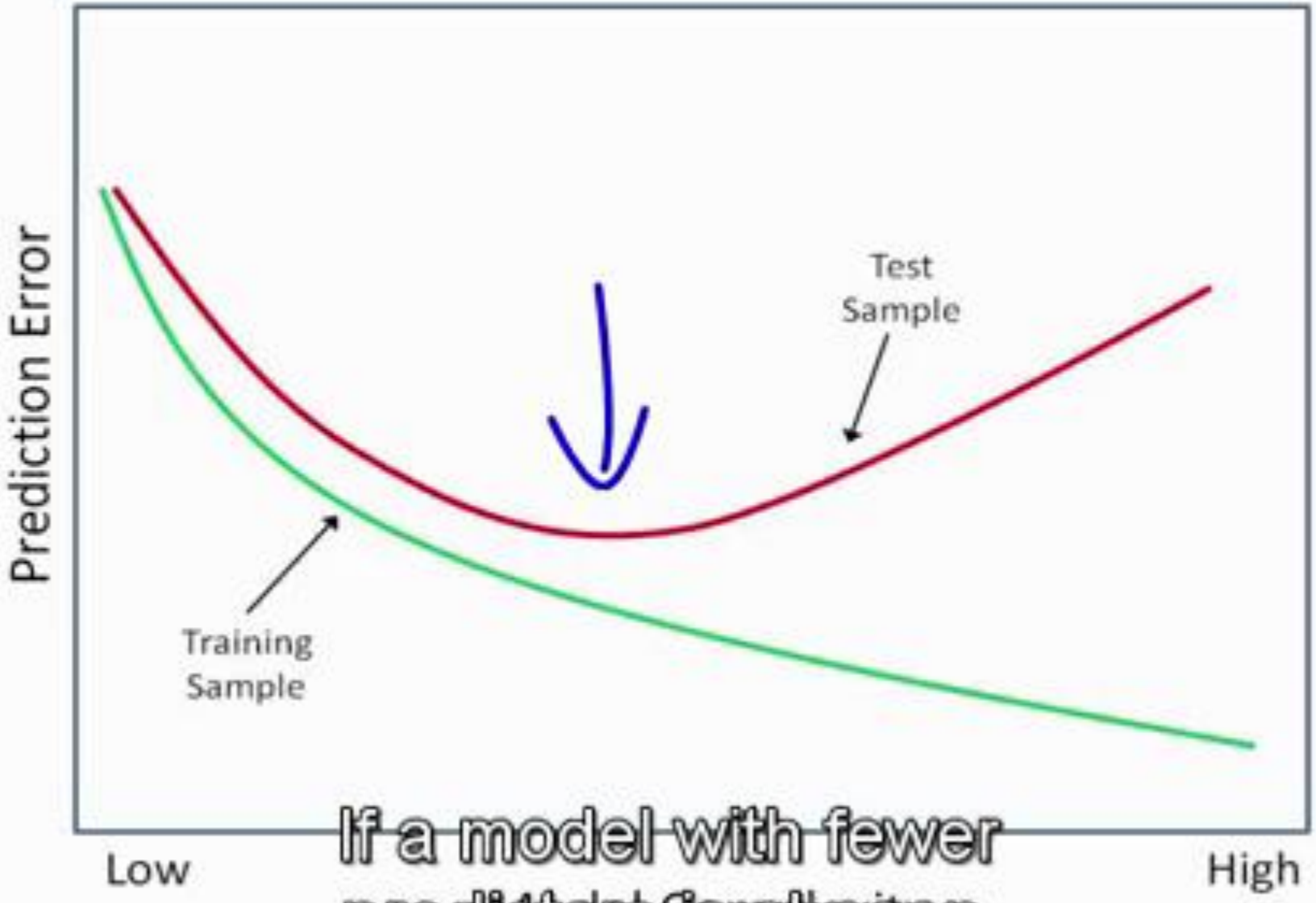
If we'd had even more predictors in our example to predict school connectiveness.



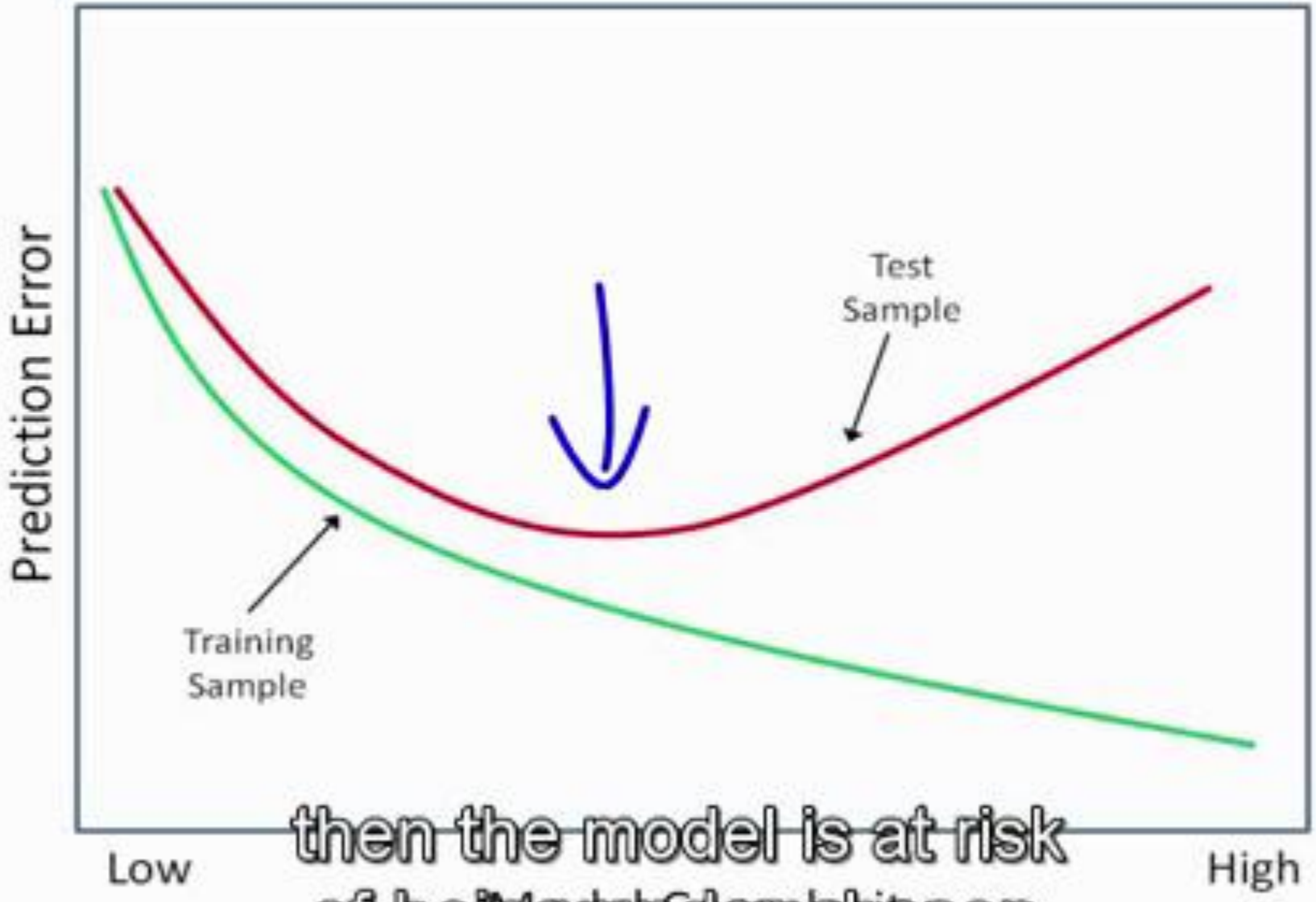
is the one that falls somewhere in here.



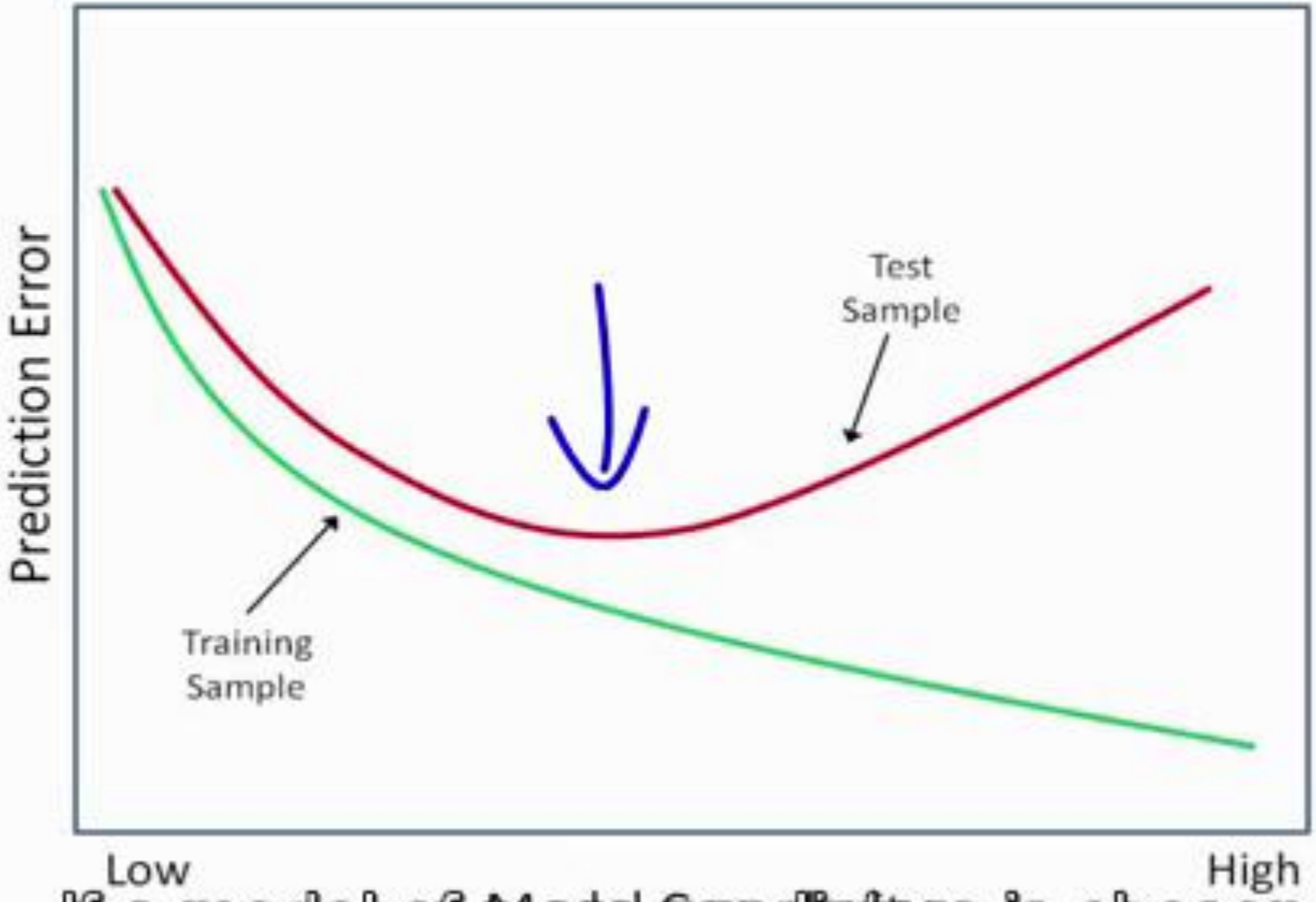
It is a point where bias and variance and the test prediction error is lowest.



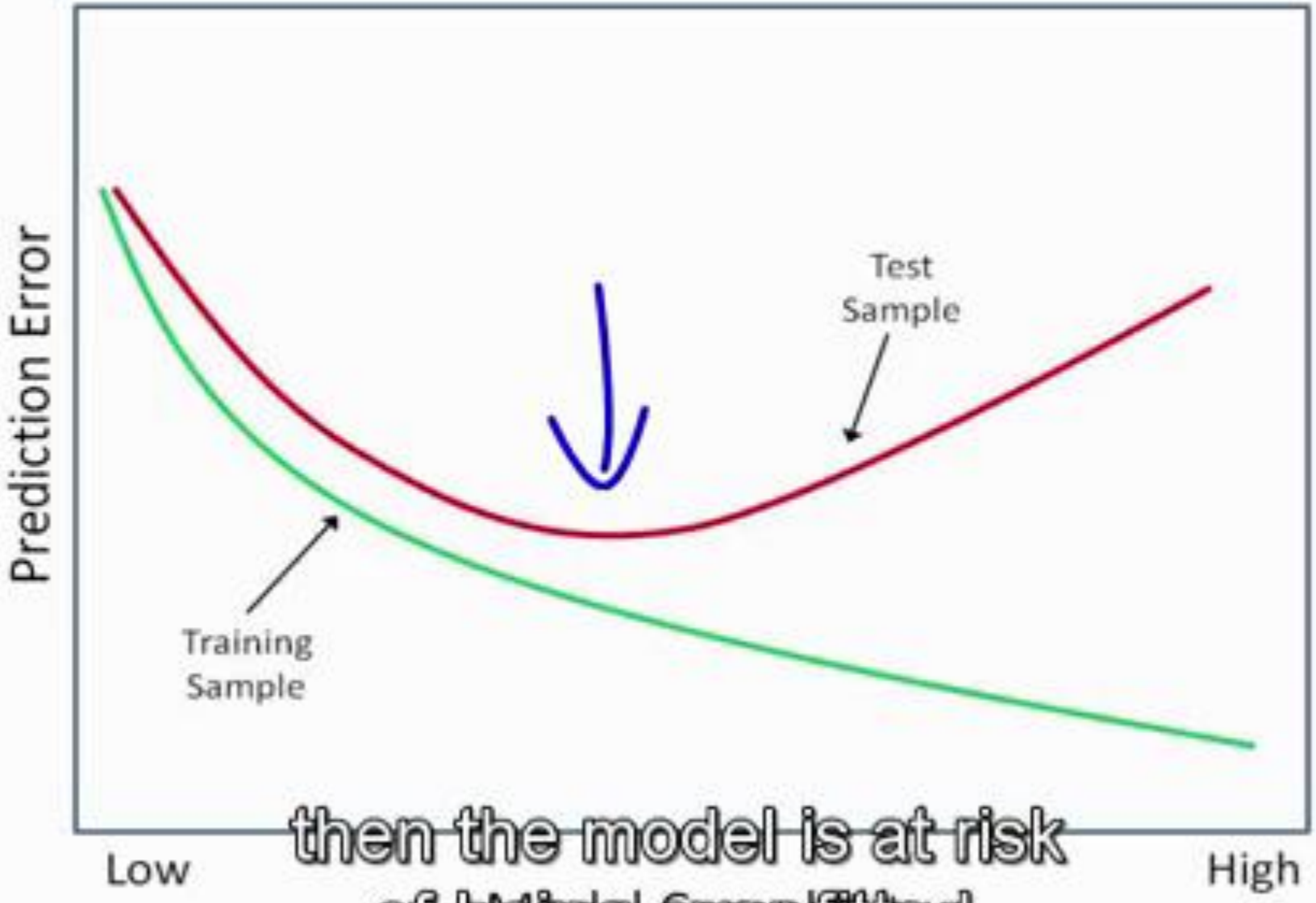
If a model with fewer
predictors is chosen,



then the model is at risk
of being under-chosen.



If a model of more predictors is chosen,



then the model is at risk
of being over-fitted.

Module 3
Lesson 3 - Lasso Regression Limitations

(c) Wesleyan University, 2016

1. Selection of variables is 100% statistically driven
2. If predictors are correlated, lasso arbitrarily selects one
3. Estimating p-values is not straightforward
4. Different selection methods, statistical software can provide different results
5. No guarantee that selected model is <u>not</u> overfitted, nor that it's the best model

Without human intervention,
variable selection methods

could produce models that make little
sense or that turn out to be useless for

accurately predicting a response
variable in the real world.

So if you find yourself in a position
which you have a large number of potential

predictors of a response variable,
what do you do?

22+X=Y
Just throwing them all in a regression
model won't work out very well.

So you need a way to identify a subset
of the most important predictors.

The best solution may be a combination of
machine learning, human intervention, and

independent application.

You could use a penalized regression
method, like Lasso Regression.

then see how well it predicts
the response variable in a test data set.

To select the subset of variables
using cross validation and

then see how well it predicts
the response variable in a test data set.

model a little to include variables
that might serve an important function.

Then you might want to tweak the
statistical model to tweak the statistical

model a little to include variables
that might serve an important function.

Despite not being directly related
to this response variable.