Regression Models with **Regularization**

Cesar Acosta Ph.D.

OVERVIEW

- Norm of a vector
- Linear Regression Loss function
- Ridge and LASSO regression
- Example: Regression with regularization
- Logistic Regression Loss function
- Example: Logistic regression with regularization

INTRODUCTION

Regression models with regularization

- Ridge regression (L2 regularization)
- LASSO regression (L1 regularization)
- Elastic net regression

Shrinkage Methods

Norm of a Vector

Shrinkage Methods

Norm of a vector → a measure of the length of a vector

Vector $\underline{b}' = [\, b_1, \ldots, b_m \,]$

ℓ_2 norm $\qquad \|b\|_2 = \sqrt{b_1^2 + \cdots + b_m^2}$

ℓ_1 norm $\qquad \|b\|_1 = |b_1| + \cdots + |b_m|$

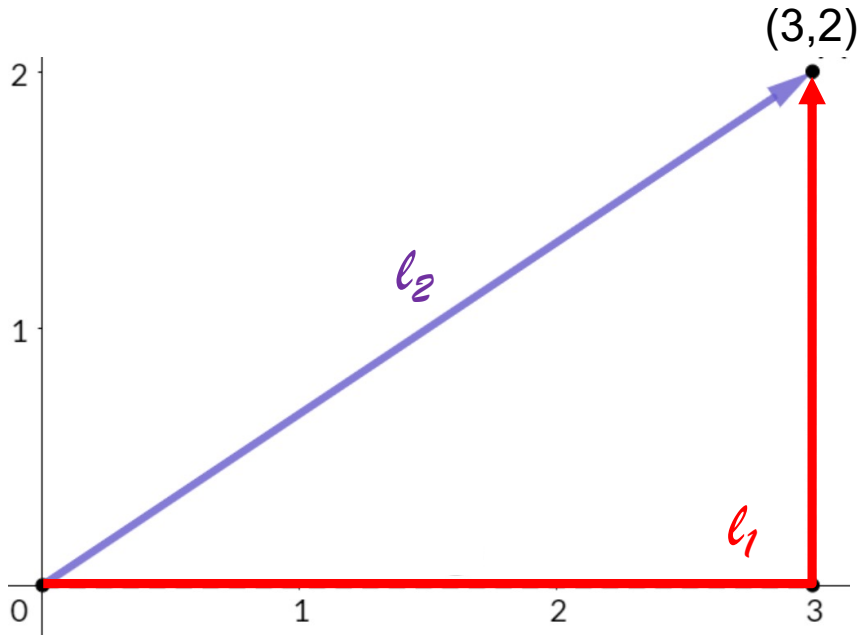
Shrinkage Methods

Norm of a vector a measure of the length of a vector

Vector $\underline{b}' = [3,2]$

ℓ_2 norm $\quad \|b\|_2 = \sqrt{3^2 + 2^2}$

ℓ_1 norm $\quad \|b\|_1 = |3| + |2|$



Ridge and LASSO Regression Models

Cesar Acosta Ph.D.

LOSS FUNCTION = COST FUNCTION

Linear Regression loss function (sum of squared errors)

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Find $b_0, \ldots, b_p$
that minimize SSE

LOSS FUNCTIONS

Linear Regression loss function

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Find $b_0, \ldots, b_p$
that minimize SSE

Ridge Regression loss function

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p} b_i^2$$

penalty

This term prevents
large $b_1, \ldots, b_p$

RIDGE REGRESSION

Ridge Regression loss function

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p} b_i^2$$

- α is the *regularization* parameter
- If α = 0 (no regularization)

RIDGE REGRESSION

Ridge Regression model

loss function

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p} b_i^2$$

with solution

$$\underline{b} = [X'X + \alpha\, I\,]^{-1} X'Y$$

LASSO REGRESSION

LASSO Regression model

loss function

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha \sum_{i=1}^{p} |b_i|$$

- This loss function prevents large regression coefficients
- If α is large, some regression coefficients are equal to zero resulting in a model with less predictors

RIDGE AND LASSO LOSS FUNCTIONS

.

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\|b\|_2^2$$

$$\underset{b_0,\ldots,b_p}{\text{Min}} \quad SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha\|b\|_1$$

Example
Baseball Players

Shrinkage Methods

- The Hitters.csv file includes data about baseball players, such as their salary and 19 player's performance measures
- To predict the player's salary we will fit regression models with regularization
- We start by removing all rows with missing values in column Salary

Shrinkage Methods

- Fit 100 ridge regression models with $10^{-2} < \alpha < 10^{10}$
- Show how the coefficients $b_1, b_2 \ldots, b_{19}$ shrink when α increases
- Find the best value for α using
  - holdout cross validation
  - 5-fold cross validation
- Use the best α value to fit a ridge regression model
- Compute the test MSE
- Repeat with LASSO regression

Ridge Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
# for holdout cv
from sklearn.model_selection import train_test_split
```

```
# for K-fold cv
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

```
from sklearn.linear_model import Ridge, Lasso
```

Ridge Regression

```
df = pd.read_csv('Hitters.csv')
df.shape
```

```
(322, 20)
```

first 16 columns

```
df[:5]
```

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts
0	293	66	1	30	29	14	1	293	66	1	30	29	14	A	E	446
1	315	81	7	24	38	39	14	3449	835	69	321	414	375	N	W	632
2	479	130	18	66	72	76	3	1624	457	63	224	266	263	A	W	880
3	496	141	20	65	78	37	11	5628	1575	225	828	838	354	N	E	200
4	321	87	10	39	42	30	2	396	101	12	48	46	33	N	E	805

Ridge Regression

```
df = pd.read_csv('Hitters.csv')
df.shape
```

```
(322, 20)
```

last 15 columns

```
df.iloc[:5,-15:]
```

Y

	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
0	14	1	293	66	1	30	29	14	A	E	446	33	20	NaN	A
1	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.0	N
2	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.0	A
3	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.0	N
4	30	2	396	101	12	48	46	33	N	E	805	40	4	91.5	N

```
# drop NAs
```

```
d0 = df.dropna()
d0.shape
```

```
(263, 20)
```

Ridge Regression

```
df = pd.read_csv('Hitters.csv')
df.shape
```

```
(322, 20)
```

```
df.iloc[:5,-15:]
```

	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
0	14	1	293	66	1	30	29	14	A	E	446	33	20	NaN	A
1	39	14	3449	835	69	321	414	375	N	W	632	43	10	475.0	N
2	76	3	1624	457	63	224	266	263	A	W	880	82	14	480.0	A
3	37	11	5628	1575	225	828	838	354	N	E	200	11	3	500.0	N
4	30	2	396	101	12	48	46	33	N	E	805	40	4	91.5	N

```
# drop NAs
```

```
d0 = df.dropna()
d0.shape
```

```
(263, 20)
```

Ridge Regression

```
y = d0.Salary
x0 = d0.drop(['Salary'],axis=1)
```

```
x0.dtypes
```

AtBat	int64
Hits	int64
HmRun	int64
Runs	int64
RBI	int64
Walks	int64
Years	int64
CAtBat	int64
CHits	int64
CHmRun	int64
CRuns	int64
CRBI	int64
CWalks	int64
League	object
Division	object
PutOuts	int64
Assists	int64
Errors	int64
NewLeague	object

Ridge Regression – One-hot Encoding with pd.get_dummies( )

```
y = d0.Salary
x0 = d0.drop(['Salary'],axis=1)
```

```
# substitute categorical cols with dummy vars

x = pd.get_dummies(x0,
                   columns = ['League','Division','NewLeague'],
                   drop_first=True)
```

Assists	Errors	League_N	Division_W	NewLeague_N
43	10	1	1	1
82	14	0	1	0
11	3	1	0	1
40	4	1	0	1

```
X = x.astype('float64')
```

```
x.dtypes
```

```
AtBat          int64
Hits           int64
HmRun          int64
Runs           int64
RBI            int64
Walks          int64
Years          int64
CAtBat         int64
CHits          int64
CHmRun         int64
CRuns          int64
CRBI           int64
CWalks         int64
PutOuts        int64
Assists        int64
Errors         int64
League_N       uint8
Division_W     uint8
NewLeague_N    uint8
```

Ridge Regression – Find 100 values in the interval $0.01 < \alpha < 10^{10}$

```
# Create a 1D array of 100 alpha values
# ranging from very small to very large
```

```
alphas = 10**np.linspace(10,-2,100)
```

split interval (10,-2) into 100 subintervals

```
alphas.shape
```

```
(100,)
```

```
# 10^(-2)
alphas.min()
```

```
0.01
```

```
# 10^10
alphas.max()
```

```
10000000000.0
```

```
# fit 100 Ridge regression models,
# one for each alpha (scaling all cols)
```

```
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

```
model = Ridge()
```

```
coefs = []
for a in alphas:
    model.set_params(alpha = a)
    model.fit(X_scaled, y)
    coefs.append(model.coef_)
```

coefs is a list of 1D arrays (vectors)
The arrays have the regression coefficients

Ridge Regression – Find 100 values in the interval $0.01 < \alpha < 10^{10}$

```
# Create a 1D array of 100 alpha values
# ranging from very small to very large
```

```
alphas = 10**np.linspace(10,-2,100)
```

split interval (10,-2) into 100 subintervals

```
alphas.shape
```

```
(100,)
```

```
# 10^(-2)
alphas.min()
```

```
0.01
```

```
# 10^10
alphas.max()
```

```
10000000000.0
```

```
# fit 100 Ridge regression models,
# one for each alpha (scaling all cols)
```

```
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)
```

```
model = Ridge()
```

```
coefs = []
for a in alphas:
    model.set_params(alpha = a)
    model.fit(X_scaled, y)
    coefs.append(model.coef_)
```

```
# Store regression coeffs
```

```
df = pd.DataFrame(coefs)
df.columns = x.columns
df.index = alphas
df.index.name = 'alpha'
```

Each row is a Ridge regression model with 19 beta coefficients

- DataFrame with ridge regression coefficients

	predictors					
	PutOuts	Assists	Errors	League_N	Division_W	NewLeague_N
alpha						
1.000000e+10	0.000	0.000	-0.000	-0.000	-0.000	-0.000
7.564633e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
5.722368e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
4.328761e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
3.274549e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
...	...	...	...	...	...	...
3.053856e-02	78.762	53.500	-22.255	31.264	-58.504	-12.455
2.310130e-02	78.762	53.557	-22.233	31.261	-58.482	-12.430
1.747528e-02	78.762	53.600	-22.216	31.258	-58.466	-12.411
1.321941e-02	78.762	53.632	-22.203	31.256	-58.453	-12.396
1.000000e-02	78.761	53.657	-22.193	31.255	-58.444	-12.385

100 rows x 19 columns

Each row is a Ridge regression model with 19 beta coefficients

- DataFrame with ridge regression coefficients

model 1

model 100

	predictors					
	PutOuts	Assists	Errors	League_N	Division_W	NewLeague_N
alpha						
1.000000e+10	0.000	0.000	-0.000	-0.000	-0.000	-0.000
7.564633e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
5.722368e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
4.328761e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
3.274549e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
...	...	...	...	...	...	...
3.053856e-02	78.762	53.500	-22.255	31.264	-58.504	-12.455
2.310130e-02	78.762	53.557	-22.233	31.261	-58.482	-12.430
1.747528e-02	78.762	53.600	-22.216	31.258	-58.466	-12.411
1.321941e-02	78.762	53.632	-22.203	31.256	-58.453	-12.396
1.000000e-02	78.761	53.657	-22.193	31.255	-58.444	-12.385

100 rows x 19 columns

Each row is a Ridge regression model with 19 beta coefficients

- DataFrame with ridge regression coefficients

	predictors					
	PutOuts	Assists	Errors	League_N	Division_W	NewLeague_N
alpha						
1.000000e+10	0.000	0.000	-0.000	-0.000	-0.000	-0.000
7.564633e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
5.722368e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
4.328761e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
3.274549e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
...	...	...	...	...	...	...
3.053856e-02	78.762	53.500	-22.255	31.264	-58.504	-12.455
2.310130e-02	78.762	53.557	-22.233	31.261	-58.482	-12.430
1.747528e-02	78.762	53.600	-22.216	31.258	-58.466	-12.411
1.321941e-02	78.762	53.632	-22.203	31.256	-58.453	-12.396
1.000000e-02	78.761	53.657	-22.193	31.255	-58.444	-12.385

α very large

α very small

100 rows x 19 columns

Each row is a Ridge regression model with 19 beta coefficients

- DataFrame with ridge regression coefficients

α very large

	predictors					
	PutOuts	Assists	Errors	League_N	Division_W	NewLeague_N
alpha						
1.000000e+10	0.000	0.000	-0.000	-0.000	-0.000	-0.000
7.564633e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
5.722368e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
4.328761e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
3.274549e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
...	...	...	...	...	...	...
3.053856e-02	78.762	53.500	-22.255	31.264	-58.504	-12.455
2.310130e-02	78.762	53.557	-22.233	31.261	-58.482	-12.430
1.747528e-02	78.762	53.600	-22.216	31.258	-58.466	-12.411
1.321941e-02	78.762	53.632	-22.203	31.256	-58.453	-12.396
1.000000e-02	78.761	53.657	-22.193	31.255	-58.444	-12.385

100 rows x 19 columns

Each row is a Ridge regression model with 19 beta coefficients

- DataFrame with ridge regression coefficients
- How does each ridge regression coefficient changes with alpha?

19 predictors

alpha	PutOuts	Assists	Errors	League_N	Division_W	NewLeague_N
1.000000e+10	0.000	0.000	-0.000	-0.000	-0.000	-0.000
7.564633e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
5.722368e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
4.328761e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
3.274549e+09	0.000	0.000	-0.000	-0.000	-0.000	-0.000
...	...	...	...	...	...	...
3.053856e-02	78.762	53.500	-22.255	31.264	-58.504	-12.455
2.310130e-02	78.762	53.557	-22.233	31.261	-58.482	-12.430
1.747528e-02	78.762	53.600	-22.216	31.258	-58.466	-12.411
1.321941e-02	78.762	53.632	-22.203	31.256	-58.453	-12.396
1.000000e-02	78.761	53.657	-22.193	31.255	-58.444	-12.385

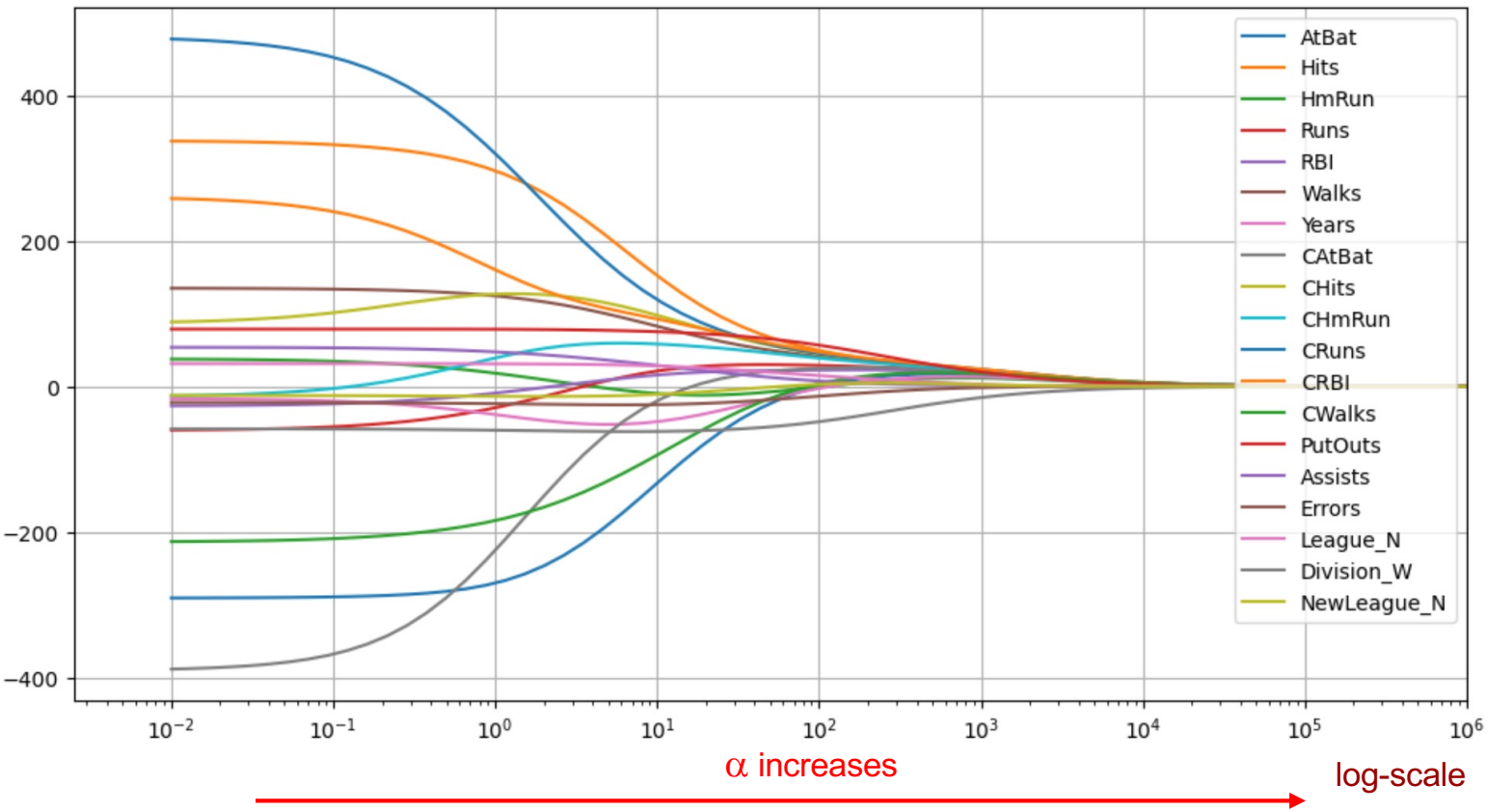
100 rows x 19 columns

Ridge Regression

There are 19 curves one for each predictor

Each curve shows how the value of a ridge regression coefficient changes when α increases

```
df.plot(figsize=(12,6),grid=True,logx=True,xlim = (0.00,10**6))
```

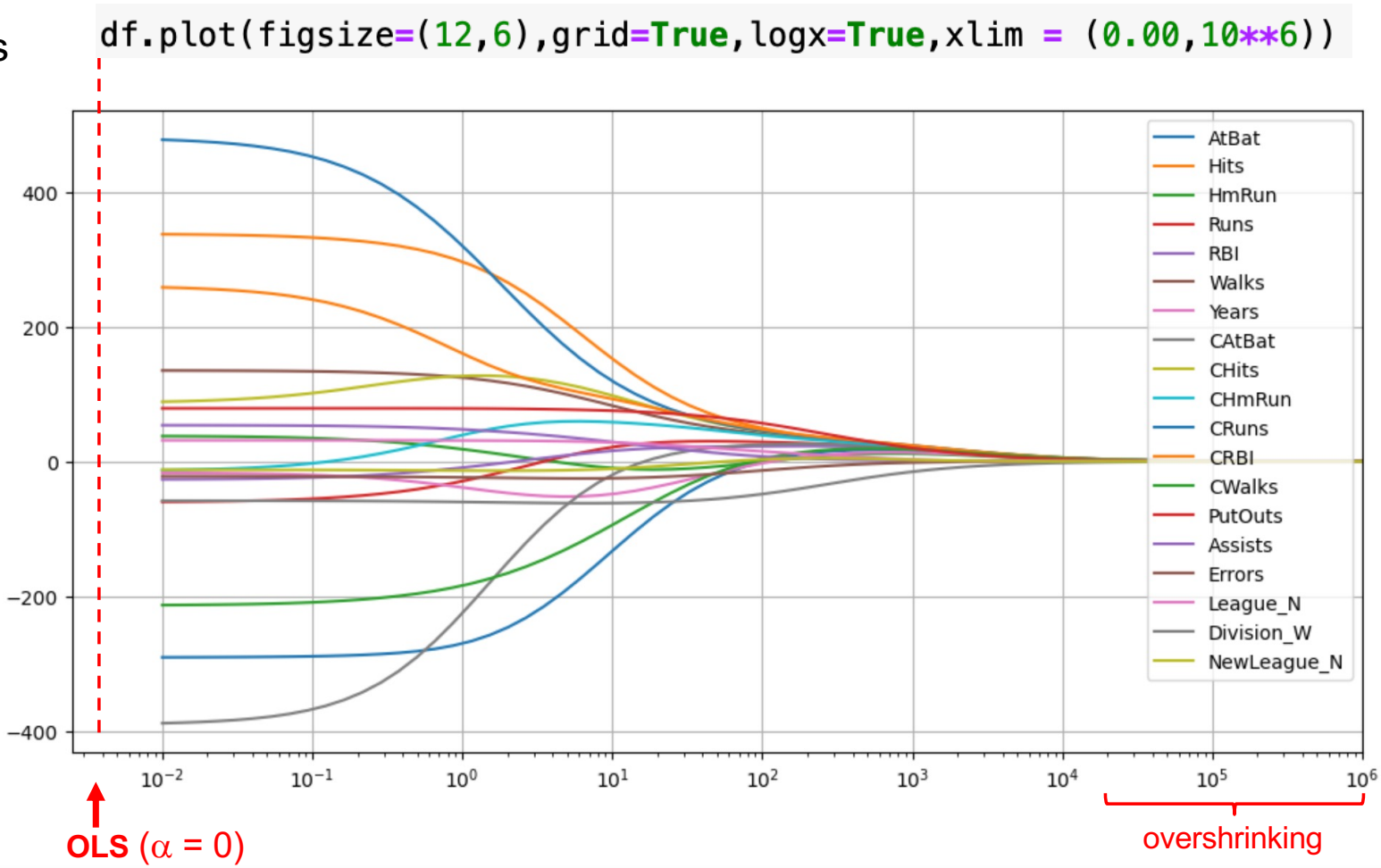


Ridge Regression

There are 19 curves one for each predictor

Each curve shows how the value of a ridge regression coefficient changes when α increases

All coefficients *shrink* to zero as alpha increases



Holdout Cross Validation

Ridge Regression – Holdout Cross Validation with fixed alpha

```
X_train,X_test,y_train,y_test = train_test_split(X,y,
                                                 test_size=0.5,
                                                 random_state=1)
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

fit model with alpha = 4

```
ridge2 = Ridge(alpha=4)
ridge2.fit(X_train_scaled, y_train)
pred2 = ridge2.predict(X_test_scaled)
mspe = mean_squared_error(y_test, pred2)
mspe
```

```
102144.52395076505
```

fit model with huge alpha = 10^9

```
ridge3 = Ridge(alpha=10**9)
ridge3.fit(X_train_scaled, y_train)
pred3 = ridge3.predict(X_test_scaled)
mspe = mean_squared_error(y_test, pred3)
mspe
```

```
172862.0826375247
```

Huge alpha makes coefficients very close to zero which increases MSPE

Ridge Regression – Holdout Cross Validation – Comparing test MSE

Linear Regression (alpha = 0)

```
ols_model = Ridge(alpha=0)
ols_model.fit(X_train_scaled, y_train)
pred = ols_model.predict(X_test_scaled)
ols_mse = mean_squared_error(y_test, pred)
ols_mse
```

```
116690.468566612
```

fit model with alpha = 4

```
ridge2 = Ridge(alpha=4)
ridge2.fit(X_train_scaled, y_train)
pred2 = ridge2.predict(X_test_scaled)
mspe = mean_squared_error(y_test, pred2)
mspe
```

```
102144.52395076505
```

fit model with huge alpha = 10^9

```
ridge3 = Ridge(alpha=10**9)
ridge3.fit(X_train_scaled, y_train)
pred3 = ridge3.predict(X_test_scaled)
mspe = mean_squared_error(y_test, pred3)
mspe
```

```
172862.0826375247
```

Huge alpha makes coefficients very close to zero which increases MSPE

Ridge Regression – Holdout Cross Validation searching for alpha

```
X_nontest,X_test,y_nontest,y_test = train_test_split(X,y,
                                                     test_size=0.40,
                                                     random_state=1)

X_train,X_validation,y_train,y_validation = train_test_split(X_nontest,y_nontest,
                                                     test_size=0.5,
                                                     random_state=1)
```

Ridge Regression – Holdout Cross Validation searching for alpha

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_validation_scaled = scaler.transform(X_validation)
```

```
ridge_model = Ridge()
```

```
validation_mspes = []

for a in alphas:
    ridge_model.set_params(alpha = a)
    ridge_model.fit(X_train_scaled, y_train)
    yhat = ridge_model.predict(X_validation_scaled)
    mspe = mean_squared_error(y_validation,yhat)
    validation_mspes.append(mspe)
```

```
df = pd.DataFrame(validation_mspes,
                  columns = ['Valid_MSE'])
df.index = alphas
df.index.name = 'alpha'
df
```

alpha	Valid_MSE
1.000000e+10	161825.790184
7.564633e+09	161825.786925
5.722368e+09	161825.782617
4.328761e+09	161825.776922
3.274549e+09	161825.769393
.	
.	
.	
0.000000	116690.468566 OLS

Ridge Regression – Holdout Cross Validation searching for alpha

```
df = pd.DataFrame(validation_mspes,
                  columns = ['Valid_MSE'])
df.index = alphas
df.index.name = 'alpha'
df
```

alpha	Valid_MSE
1.000000e+10	161825.790184
7.564633e+09	161825.786925
5.722368e+09	161825.782617
4.328761e+09	161825.776922
3.274549e+09	161825.769393
...	...
3.053856e-02	111339.084798
2.310130e-02	111577.207548
1.747528e-02	111800.332935
1.321941e-02	112003.332742
1.000000e-02	112182.439807

```
min1 = df.Valid_MSE.min()
min1
```

```
78821.05978981467
```

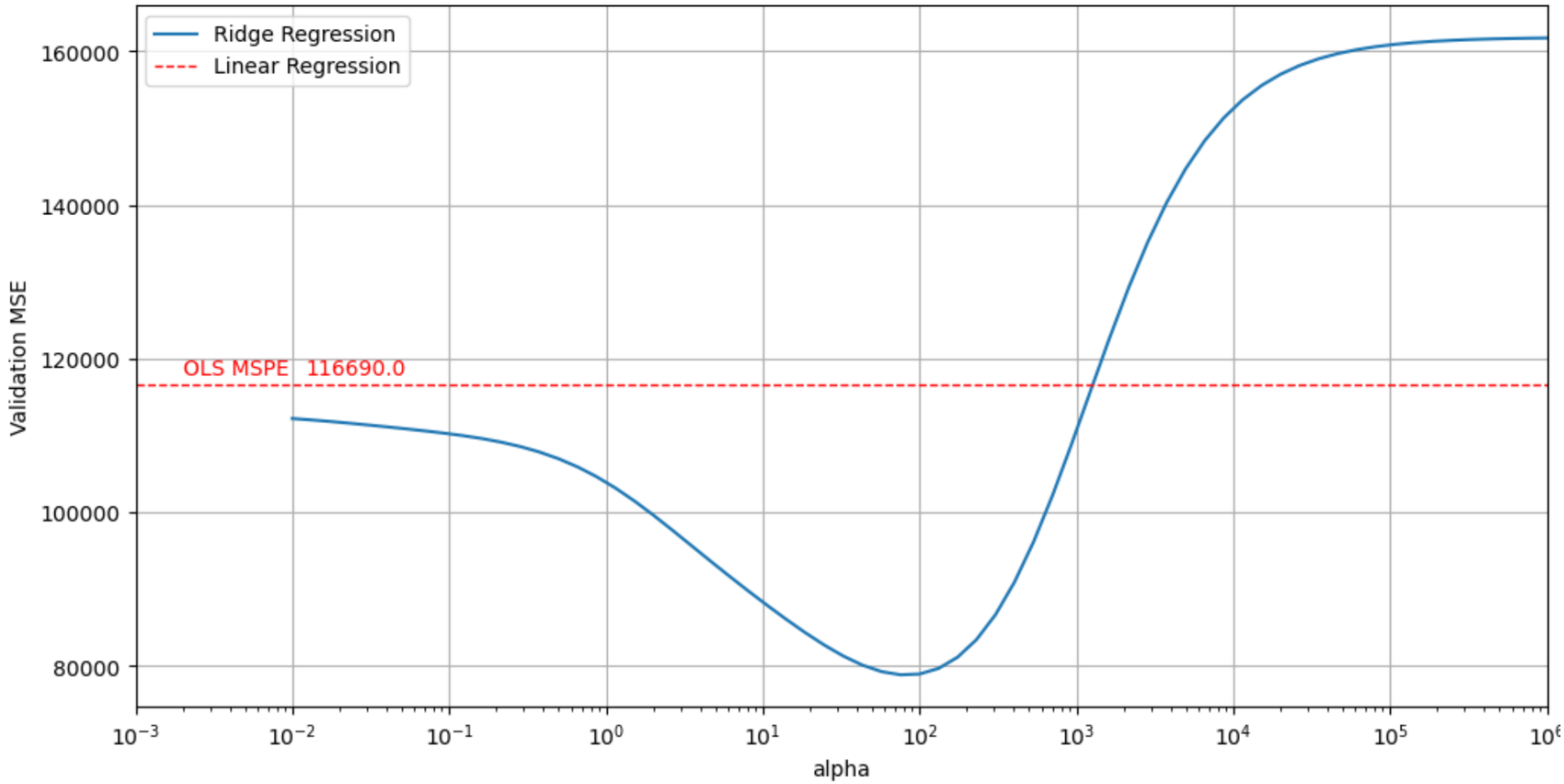
```
df[df.Valid_MSE == min1]
```

alpha	Valid_MSE
75.646333	78821.05979

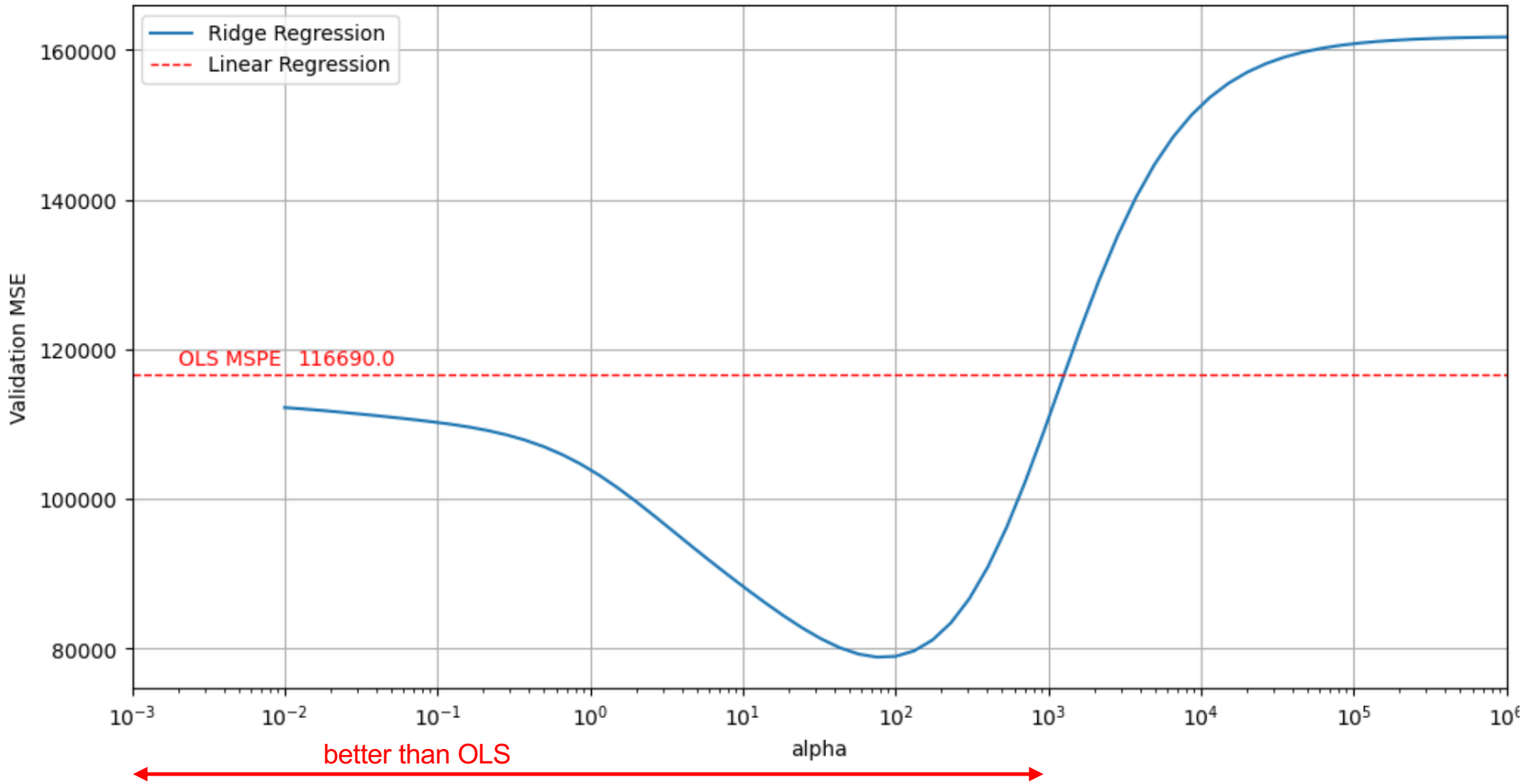
```
# best alpha giving min Validation MSE
best_alpha = df.Valid_MSE.idxmin()
best_alpha
```

```
75.64633275546291
```

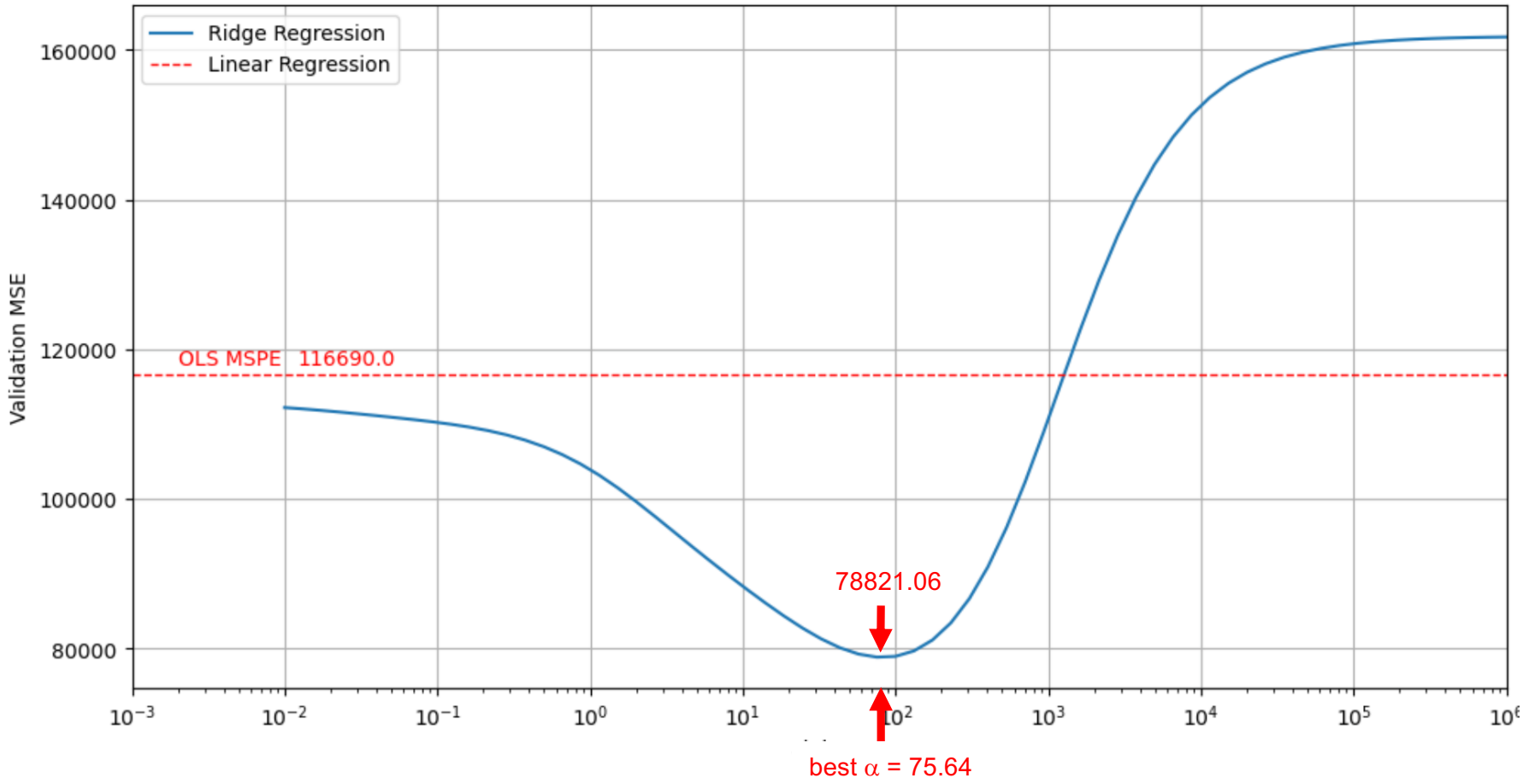
Ridge Regression – Validation Approach



Ridge Regression Models better than OLS



Ridge Regression best Model



Ridge Regression – Test MSE from Holdout CV

```
min1 = df.Valid_MSE.min()
min1
```

```
78821.05978981467
```

← Validation MSE

```
df[df.Valid_MSE == min1]
```

alpha	Valid_MSE
75.646333	78821.05979

```
# best alpha giving min Validation MSE
best_alpha = df.Valid_MSE.idxmin()
best_alpha
```

```
75.64633275546291
```

```
scaler = StandardScaler()
scaler.fit(X_nontest)
X_nontest_scaled = scaler.transform(X_nontest)
X_test_scaled = scaler.transform(X_test)
```

```
ridge2 = Ridge(alpha = best_alpha)
ridge2.fit(X_nontest_scaled, y_nontest)
pred2 = ridge2.predict(X_test_scaled)
mspe = mean_squared_error(y_test, pred2)
mspe
```

```
99508.83201629658
```

← Test MSE

5-fold Cross Validation

Ridge Regression 5-fold cross validation to find best alpha

```
X_train,X_test,\
y_train,y_test = train_test_split(X,y,test_size=0.5,
                                  random_state=1)
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
scaler = StandardScaler()
model = Ridge()
pipe1 = Pipeline([('scaler', scaler),('ridge', model)])
```

```
param_grid = {'ridge__alpha': alphas}
```

```
ridgecv = GridSearchCV(pipe1,param_grid,cv = 5,
                       scoring = 'neg_mean_squared_error')
ridgecv.fit(X_train, y_train);
```

Ridge Regression 5-fold cross validation to find best alpha

```
X_train,X_test,\
y_train,y_test = train_test_split(X,y,test_size=0.5,
                                  random_state=1)
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
scaler = StandardScaler()
model = Ridge()
pipe1 = Pipeline([('scaler', scaler),('ridge', model)])
```

```
param_grid = {'ridge__alpha': alphas}
```

```
ridgecv = GridSearchCV(pipe1,param_grid,cv = 5,
                       scoring = 'neg_mean_squared_error')
ridgecv.fit(X_train, y_train);
```

```
# best alpha (minimizing validation MSE)
```

```
ridgecv.best_params_
```

```
{'ridge__alpha': 100.0}
```

```
alpha1 = ridgecv.best_params_['ridge__alpha']
alpha1
```

```
100.0
```

```
# test MSE with best alpha
```

```
# fit RR model with best alpha
ridge1 = Ridge(alpha=alpha1)
ridge1.fit(X_train_scaled,y_train);
```

```
yhat = ridge1.predict(X_test_scaled)
best_mspe = mean_squared_error(y_test, yhat)
best_mspe
```

```
99586.56834382795
```

← Test MSE

PREDICTION

```
# fit RR model with best alpha
ridge1 = Ridge(alpha=alpha1)
ridge1.fit(X_train_scaled,y_train);
```

```
# coefficients of best RR model

df4 = pd.DataFrame(ridge1.coef_,index=X.columns,
                   columns=['ridge_coeff'])
df4
```

	ridge_coeff		
AtBat	3.911359	CHmRun	41.512925
Hits	36.096360	CRuns	33.271936
HmRun	1.736680	CRBI	41.571334
Runs	19.611659	CWalks	25.535833
RBI	32.219132	PutOuts	75.761366
Walks	43.972410	Assists	-2.475953
Years	8.496447	Errors	-0.792667
CAtBat	17.992839	League_N	8.214298
CHits	32.545056	Division_W	-41.608368
CHmRun	41.512925	NewLeague_N	5.211955

PREDICTION

predict salary of first player in test set

```
X_test[:1]
```

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks
126	282.0	78.0	13.0	37.0	51.0	29.0	5.0	1649.0	453.0	73.0	211.0	280.0	138.0

```
newval = X_test_scaled[:1]
newval
```

```
array([[-0.75403349, -0.59086622,  0.20177987, -0.63314908, -0.00973381,
        -0.45405721, -0.50193827, -0.42197414, -0.40314783,  0.01162816,
        -0.43551916, -0.17330772, -0.4452582 ,  1.27997267, -0.33878194,
        -0.44171974, -1.03892496,  0.99239533, -1.00766295]])
```

```
ridge1.predict(newval)
```

Predicted Salary →

```
array([461.60866494])
```

```
y_test[:1]
```

Actual Salary →

```
126    500.0
Name: Salary, dtype: float64
```