HW3

February 9, 2020

[1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

[2]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest
from sklearn.metrics import mean_squared_error
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
from sklearn.linear_model import ElasticNetCV
```

[3]:
```
np.random.seed(2)
```

1 Training/test error for subset selection

1.1 1.1

[90]:
```
X=np.random.normal(size=(1000, 20))

eps=np.random.normal(0.0,1.0,1000)

# replace 4 beta term to zero
beta=np.random.randint(-10,10,20)
index=np.random.randint(1, 20, 4)
beta[index]=0

y=np.matmul(X, beta)+eps
```

[91]:
```
beta
```

[91]:
```
array([  5,   8,   0,  -7,  -7,  -2,   7,   7,   0,   0,  -3,   9, -10,
         8,   8,   7,  -5,  -6,   0,  -6])
```

[92]:
```
y.shape
```

[92]: (1000,)

1.2 1.2

[93]:
```
x_train, x_test, y_train, y_test= train_test_split(X,y,train_size=100)
```

[94]:
```
x_train.shape
```

[94]: (100, 20)

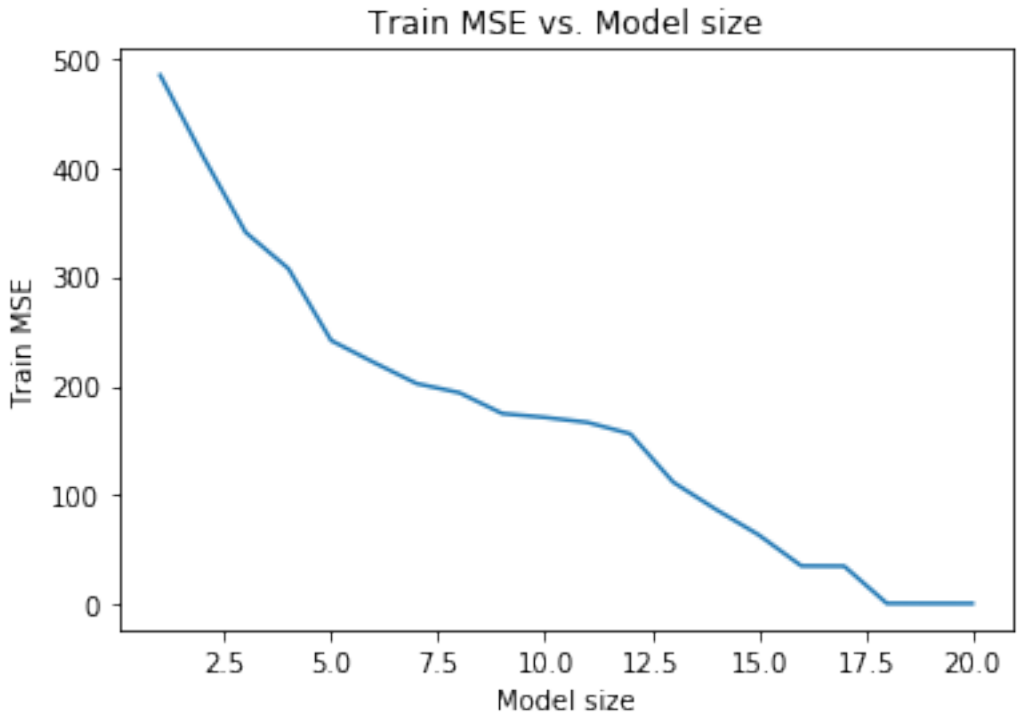
1.3 1.3

[108]:
```
# best subset selection
def best_k_subset(k):

    # select k parameters
    selector=SelectKBest(f_regression, k)
    selector.fit(x_train, y_train)
    index_selected=selector.get_support()

    # training set MSE
    x_train_new=x_train[:,index_selected]
    reg=LinearRegression().fit(x_train_new, y_train)
    y_train_pred=reg.predict(x_train_new)
    train_mse=mean_squared_error(y_train, y_train_pred)

    # test MSE
    x_test_new=x_test[:,index_selected]
    y_test_pred=reg.predict(x_test_new)
    test_mse=mean_squared_error(y_test, y_test_pred)


    return train_mse, test_mse, reg, index_selected
```

[109]:
```
# draw Train MSE
train_mse_ls=[]
test_mse_ls=[]
reg_ls=[]
index_ls=[]
for i in range(20):
    train_mse, test_mse, reg, index_selected=best_k_subset(i+1)
    train_mse_ls.append(train_mse)
    test_mse_ls.append(test_mse)
    reg_ls.append(reg)
    index_ls.append(index_selected)

plt.plot(np.arange(1,21), train_mse_ls)
```

```
plt.xlabel("Model size")
plt.ylabel("Train MSE")
plt.title("Train MSE vs. Model size")
```

[109]: Text(0.5, 1.0, 'Train MSE vs. Model size')



[102]:
```
min_mse=min(train_mse_ls)
index_mse=train_mse_ls.index(min(train_mse_ls))+1
print(f"The min MSE is {min_mse} from model size {index_mse}.")
```

```
The min MSE is 0.846569155563872 from model size 20.
```

1.4 1.4 + 1.5

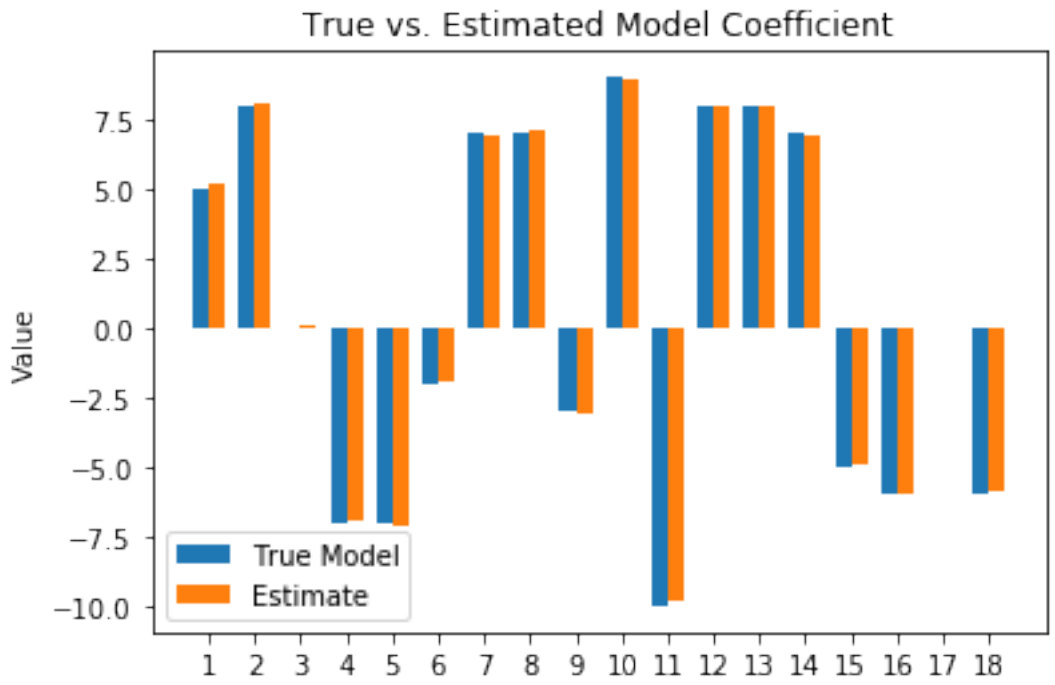
[97]:
```
# draw test MSE
plt.plot(np.arange(1,21), test_mse_ls)
plt.xlabel("Model size")
plt.ylabel("Test MSE")
plt.title("Test MSE vs. Model size")

min_mse=min(test_mse_ls)
index_mse=test_mse_ls.index(min(test_mse_ls))+1
print(f"The min MSE is {min_mse} from model size {index_mse}.")
```

```
The min MSE is 1.1072266766793832 from model size 18.
```

1.5 1.6

[116]:
```
# at model with 18 parameters the Test MSE is minimized
best_reg=reg_ls[17]
best_reg.coef_

labels = np.arange(1,19)
x = np.arange(len(labels))  # the label locations
width = 0.35  # the width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, beta[index_ls[17]], width, label='True Model')
rects2 = ax.bar(x + width/2, best_reg.coef_, width, label='Estimate')

ax.set_ylabel('Value')
ax.set_title('True vs. Estimated Model Coefficient')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
```

[116]:
```
<matplotlib.legend.Legend at 0x1a22b90310>
```



Based on the coefficient size, we could see that the coefficients of the estimated model are very close to the true model which generates the data.

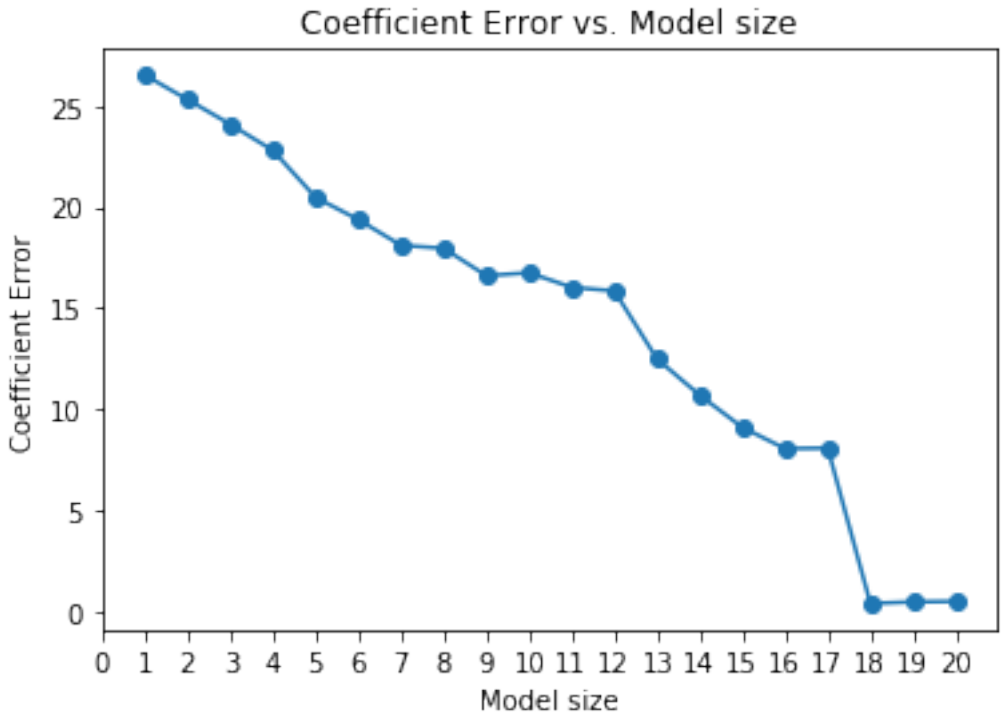
1.6 1.7

[134]:
```
def cal_beta_error(r):
    estimate=reg_ls[r].coef_
    index_mask=index_ls[r]
    # for selected features
    beta_slice=beta[index_mask]
    inner_err=(beta_slice- estimate)**2
    # for other features
    invert_mask=np.invert(index_mask)
    beta_rest=beta[invert_mask]
    inner_err2=np.square(beta_rest)
    outer_err= np.sqrt(np.sum(inner_err)+np.sum(inner_err2))
    return outer_err
```

[138]:
```
beta_err_ls=[]
for i in range(20):
    beta_err=cal_beta_error(i)
    beta_err_ls.append(beta_err)

plt.plot(np.arange(1,21), beta_err_ls, marker='o')
plt.xticks(np.arange(0,21))
```

```
plt.xlabel("Model size")
plt.ylabel("Coefficient Error")
plt.title("Coefficient Error vs. Model size")
```

[138]:
```
Text(0.5, 1.0, 'Coefficient Error vs. Model size')
```



Just like the test MSE plot, the coefficient error drops as the number of selected model features increases. It is worth noting that, around the best subset (18 parameters), the coefficient error drops sharply. This findings further validate the subset selection strategy.

2 2. Application exercises

[141]:
```
gss_train=pd.read_csv("./data/gss_train.csv")
gss_test=pd.read_csv("./data/gss_test.csv")
```

[148]:
```
x_train=gss_train.drop(['egalit_scale'], axis=1)
y_train=gss_train['egalit_scale']
x_test=gss_test.drop(['egalit_scale'], axis=1)
y_test=gss_test['egalit_scale']
```

2.1 2.1

[149]:
```
# OLS
ols=LinearRegression().fit(x_train, y_train)
y_pred=ols.predict(x_test)
mse=mean_squared_error(y_test, y_pred)
print('The Test MSE for Least square linear model is: ', mse)
```

```
The Test MSE for Least square linear model is:  63.21362962301499
```

2.2 2.2

[150]:
```
# ridge
ridge=RidgeCV(cv=10).fit(x_train, y_train)
y_pred=ridge.predict(x_test)
mse=mean_squared_error(y_test, y_pred)
print('The Test MSE for Ridge Regression is: ', mse)
```

```
The Test MSE for Ridge Regression is:  62.4992024395781
```

2.3 2.3

[158]:
```
# lasso
lasso=LassoCV(cv=10).fit(x_train, y_train)
y_pred=lasso.predict(x_test)
mse=mean_squared_error(y_test, y_pred)
print('The Test MSE for Lasso Regression is: ', mse)
```

```
The Test MSE for Lasso Regression is:  62.7780157899344
```

[161]:
```
non_zero_coef=len(lasso.coef_)-list(lasso.coef_).count(0)
print('The number of non-zero coefficient is ', non_zero_coef)
```

```
The number of non-zero coefficient is  24
```

2.4 2.4

[163]:
```
# elastic net
e_net=ElasticNetCV(alphas=np.arange(0,1.1, 0.1), cv=10).fit(x_train, y_train)
y_pred=e_net.predict(x_test)
mse=mean_squared_error(y_test, y_pred)

non_zero_coef=len(e_net.coef_)-list(e_net.coef_).count(0)

print("alpha: ", e_net.alpha_)
print('The number of non-zero coefficient is ', non_zero_coef)
print('The Test MSE for Elastic net Regression is: ', mse)
```

```
alpha:  0.1
The number of non-zero coefficient is  40
The Test MSE for Elastic net Regression is:  62.5070860872212

/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36707.798962216766, tolerance: 12.395240465465465
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36643.571300954776, tolerance: 12.501906826706676
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36657.592109299265, tolerance: 12.28351822955739
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36732.8705848919, tolerance: 12.346995798949736
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36977.45386687045, tolerance: 12.41885041260315
```

```
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 34582.25667175855, tolerance: 12.047380195048763
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36066.44710004871, tolerance: 12.300672318079519
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36544.0116067999, tolerance: 12.143343435858965
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 37142.410690795645, tolerance: 12.340121830457614
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: UserWarning: Coordinate
descent with alpha=0 may lead to unexpected results and is discouraged.
  tol, rng, random, positive)
/Users/ziwenchen/opt/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/coordinate_descent.py:471: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Duality gap: 36925.249764764965, tolerance: 12.506681020255064
  tol, rng, random, positive)
```

3 2.5

Generally speaking, the above regression models have similar performance in prediciting egalitarian level. Every model has a MSE around 62~63, with linear model slightly higher.

Considering that the numerical range of individual's egalitarian value is from 1 to 35, the general prediction performance is OK.