GSE 524 Assignment: Linear Model Selection

The file bestsubset.py defines a function called bestsubset that takes two arguments, a dataframe X and array y. If X has k columns, then there are k variables under consideration to include in our model of y=f(X). We only consider linear models in this assignment, and the function bestsubset uses 5-folds cross validation to compare every possible linear model that includes one or more of the predictors in X. The function returns a list with two elements: a list of column indices included in the best model, and an array of coefficient estimates for that model. You should assume that X does NOT have a column for intercept, so you should have LinearRegression fit the intercept (i.e. use default setting).

For example, if X has 3 columns (x0, x1, and x2) then the models to consider are:

y = a + b x0 + e

y = a + b x1 + e

y = a + b x2 + e

y = a + b x0 + c x1 + e

y = a + b x1 + c x2 + e

y = a + b x0 + c x2 + e

y = a + b x0 + c x1 + d x2 + e

where a,b,c,d are coefficients to be estimated and e is the error term.

As an example, suppose the best model, based on the 5-folds cross validation, is y = a + b x0 + c x2 + e. Then, the function returns a list whose first element is [0,2] (i.e. the columns of X that are included in best model) and the second element is [b, c] (i.e. the coefficient estimates for that model).

The function in the bestsubset.py file suffers from a problem that if the number of predictors gets large, the function will be very slow. The total number of iterations of the inner loop is 2^k since there are 2^k combinations of predictors. Thus, this function is O(2^k), considerably slower than any algorithm we discussed in our unit on algorithmic efficiency.

There are two solutions to this computational issue that you will implement in this week's assignment.

1. Cross validation is relatively slow because it involves multiple estimations of the model (5 in the case of 5-folds CV). When comparing multiple models with the same number of predictors, we could instead use the training MSE. Recall that the training MSE will always be lower (better) for more flexible models but that is not an issue when comparing different models with same number of predictors.

   Rewrite the bestsubset function so that it only uses CV to compare models with different numbers of predictors. In the inner loop, use the training MSE to pick the best model with j predictors (where j=1,2,3,..,k) instead of doing cross validation. This will yield k potential models (one for each value of j). Then use 5-folds CV to pick among those k models. This algorithm is explained in the slides under Best Subset Selection.

2. Solution 1 still has 2^k iterations although it will be faster than the provided function by reducing the time spent on each iteration. Nonetheless, as k gets large, the algorithm is still slow. An alternative solution is forward subset selection. The algorithm is also presented in the slides on Model Selection. Basically, you start by finding the best 1-predictor model, using the training MSE to choose the best model. Then you consider adding exactly one of the remaining (k-1) predictors and pick the best one, again using training MSE to choose. Then you consider adding exactly one of the remaining (k-2) predictors, and so forth, until finally you get to the full model with all k predictors. At the end of this process, you will have k possible models, one for each j=1,2,3,…,k, where j is the number of predictors. Choose between the k potential models using 5-fold CV.
   Note that the loops in this solution have O(k^2) iterations, much better, because it does not try every combination. As an example, suppose x1 is the best 1-predictor model from the first iteration. Then, when choosing the best 2-predictor model, you only consider models **that include x1**, i.e. you only consider x1,x0 and x1,x2 (excluding x0,x2).

   Write a function called forwardsubset with same arguments and return values as bestsubset, but implementing the forward subset algorithm.

   Chapter 6.1 of the book Introduction to Statistical Learning is a reference for more information about both of these algorithms. (statlearning.com)

What to turn in:

Turn in a file called bestsubset.py that contains the modified bestsubset function and the forwardsubset function. The autograder must run in under 10 minutes to get credit. If you implement the algorithms as explained above, this will not be a problem, but if you run my code as is, it will not complete in 10 minutes.

Here are some more details about the code I provided to help you get started with modifying the code:

```
def bestsubset(X,y):
  mse=np.empty(0)
  cs=[]
  for k in range(X.shape[1]):
    for c in itertools.combinations(range(X.shape[1]),k+1):
      Xc=X.iloc[:,list(c)]
      scores = cross_val_score( LinearRegression(), Xc, y, cv=KFold(5,shuffle=True,random_state=0),
                  scoring="neg_mean_squared_error")
      mse=np.append(mse,np.mean(scores))
      cs.append(c)
  i=np.argmax(mse)
  Xc=X.iloc[:,list(cs[i])]
  model=LinearRegression()
  model.fit(Xc,y)
  model.coef_
  ret = [list(cs[i]),model.coef_]
  return ret
```

Here is a brief explanation of my code:

The outer loop (for k in range(X.shape[1]) has k take on the values 0,1,…,K-1, where K is the total number of possible predictors.

The inner loop (the one using itertools.combinations) considers every combination of predictors with exactly (k+1) predictors where (k+1) ranges from 1,…,K since k ranges from 0,…,K-1. For example, continuing the example above, when k=0, we consider the three possible 1-predictor models in the inner loop (x0, x1, and x2). When k=1, we consider the three possible 2-predictor models ((x0,x1), (x1,x2), and (x0, x2). When k=2, we consider the only 3-predictor model.

The first line in the inner loop (i.e. the one that starts: Xc=X.iloc…) sets Xc to be the correct X matrix given the chosen predictors for that iteration of the loop.

The mse array contains the “test mse” for each of the models under consideration and the cs list contains the included predictors for each model under consideration.

After the completion of the loops, we use argmax to find the model with the largest MSE (remember, cross_val_score computes negative MSE, so bigger is better), the i-th model in the list. Then we estimate model i and the function returns the coefficients from that model (model.coef_) along with the indexes of coefficients included in that model (cs[i]).