Data Analysis & Interpretation

TEXT

Monday, February 22, 2016

Machine Learning for Data Analysis: Week 3

Machine Learning for Data Analysis, Week 3: Lasso Regression

Research Introduction:

I previously selected the NESARC codebook due to personal interest in alcohol usage within the US population. After looking through the codebook for the NESARC study, I decided that I am interested in researching alcohol consumption and its association to gender, age, and socioeconomic status.

Data Preparation & Management:

For this particular assignment, running a lasso regression, bi-level categorical response and/or quantitate variables are recommended. In my previous assignments, I have used a variety of related variables that have changed throughout the courses to best suit the needs of the assignment. For this assignment, I chose a variety of bi-level categorical and quantitative explanatory (or predictor) variables and used a quantitative variable to assess the consumption of alcohol by the participant, a quantitative response, or target, variable. The variables and identification codes from the NESARC study can be found below for each of the variables used:

1. Age (AGE, Quantitative)
2. Gender (SEX, Bi-Level Categorical)
3. Full-Time Working Status (S1Q7A1, Bi-Level Categorical)
4. Total Household Income (S1Q12A, Quantitative)
5. Received Food Stamps in Last 12 Months (S1Q14A, Bi-Level Categorical)

6. Weight (S1Q24LB, Quantitative)

7. Number of Alcoholic Drinks Consumed While Drinking (S2AQ8B, Quantiative)

Gender, full-time working status, and receipt of food stamps are bi-level categorical, with values of (1) and (2) corresponding to either Male/Female or Yes/No, respectively. These were modified to ensure that predictor variables are on the same scale, with values of (1) or (0). The variables representing the participant age, total household income, weight, and number of alcoholic drinks consumed are quantitative and needed minimal data management. With provided techniques, the 'Unknown' answers, were made to be blank responses and dropped from the data set, leaving bi-level categorical and quantitative variables to utilize in the lasso regression analysis with all 'Unknown' or missing responses removed.

Statistical Results & Graphics:

Lasso regression analysis is a shrinkage and variable selection method for linear regression models. The goal of lasso regression is to obtain the subset of predictors that minimizes prediction error for a quantitative response variable. The lasso does this by imposing a constraint on the model parameters that causes regression coefficients for some variables to shrink toward zero. Variables with a regression coefficient equal to zero after the shrinkage process are excluded from the model. Variables with non-zero regression coefficients variables are most strongly associated with the response variable. A large pool of predictors was selected in order to maximize the usage of lasso regression analysis. In addition to lasso regression analysis, k-fold cross validation was used to select the best fitting model and obtain a more accurate estimate of the model's test error rate.

The lasso regression analysis was conducted to identify a subset of variables from a pool of 6 categorical and quantitative predictor variables that best predicted a quantitative response variable measuring consumption of alcoholic beverages. All predictor variables were standardized to have a mean of zero and a standard deviation of one. The data was randomly split into a training sat that included 70% of the observations and a test set that included 30% of the observation. The least angle regression algorithm (LARS) with k = 10 fold cross validation was used to estimate the lasso regression model in the training set, and the model was validated using the test set. The change in cross validation mean squared error at each step was used to identify the best subset of predictor variables.

Of the 6 predictor variables, all were retained in the selected model (meaning that none fell to zero during the estimation process). During the estimation process, age and gender were most strongly associated with consumption of alcoholic beverages, followed by total household income and weight. Age, total

household income, and full time working status were negatively associated with consumption of alcoholic beverages. Other predictors associated with positive alcohol consumption included receipt of food stamps, weight, and gender. The variable names and regression coefficients, output from the Python program, are as follows:

{‘AGE’: -0.56482336962716273,

'S1Q12A’: -0.15384448917108912,

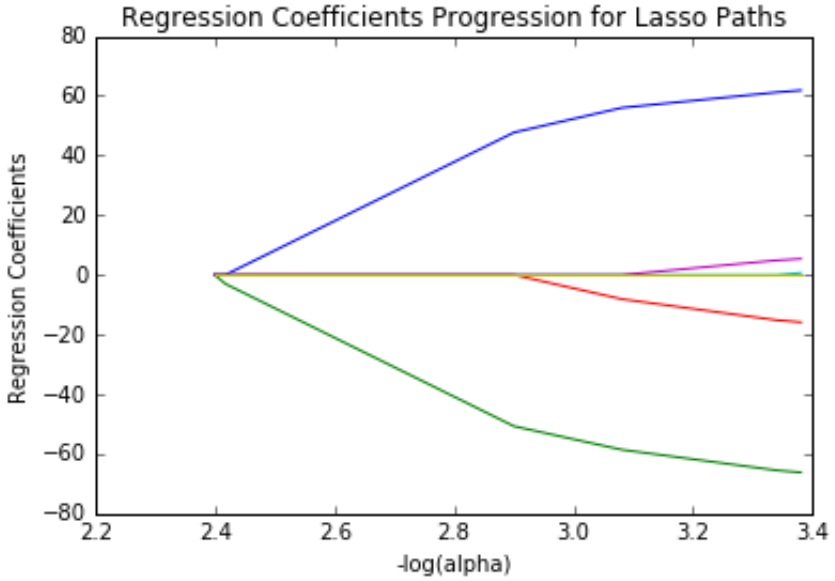
'S1Q14A’: 0.044091719929683151,

'S1Q24LB’: 0.083869296639055055,

'S1Q7A1’: -0.08129120120840401,

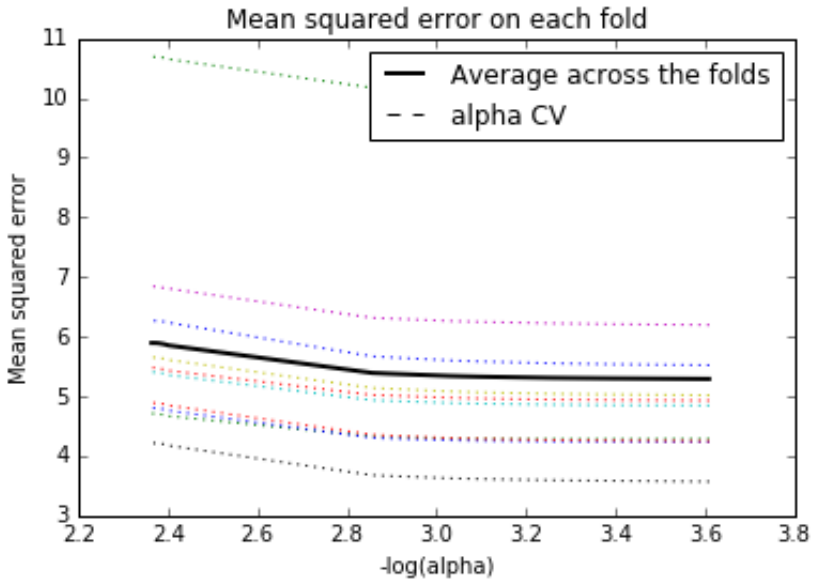
'SEX’: 0.51487389252560611}

The coefficient progression graph can be seen below:



As mentioned above, age (green) is most strongly negatively associated with the consumption of alcoholic beverages, while gender (blue) is most strongly positively associated with the consumption of alcoholic beverages. The other variables are graphed according to the Python output above, with this graph showing a visual representation of the regression coefficient progression.

The mean squared error plot for each fold can be seen below:



The mean squared error (MSE) from the training and test data sets can be seen below, along with the r-squared value from each data set, as output by the Python program:

training data MSE

5.27885387289

test data MSE

4.7479691288

training data R-square

0.104055656023

test data R-square

0.109369759037

As expected, the training data has a higher MSE than the test data set.

Python Script:

The following Python script was run to create the random forest and supporting analysis described in detail throughout this post:

```
#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


# import the entire dataset to memory

data = pd.read_csv('nesarc_pds.csv', low_memory=False)


# ensure each of these columns are numeric

data['SEX'] = data['SEX'].convert_objects(convert_numeric=True)

data['AGE'] = data['AGE'].convert_objects(convert_numeric=True)

data['S2AQ8B'] = data['S2AQ8B'].convert_objects(convert_numeric=True)

data['S1Q12A'] = data['S1Q12A'].convert_objects(convert_numeric=True)

data['S1Q14A'] = data['S1Q14A'].convert_objects(convert_numeric=True)

data['S1Q24LB'] = data['S1Q24LB'].convert_objects(convert_numeric=True)

data['S1Q7A1'] = data['S1Q7A1'].convert_objects(convert_numeric=True)


#upper-case all DataFrame column names

data.columns = map(str.upper, data.columns)


# recode missing values to python missing (NaN)

data['S2AQ8B'] = data['S2AQ8B'].replace(99, np.nan)

data['S1Q24LB'] = data['S1Q24LB'].replace(999, np.nan)

data['S2AQ8B']=data['S2AQ8B'].replace(99, np.nan)


# Data Management

data_clean = data.dropna()

recode1 = {1:1, 2:0}

data_clean['SEX']= data_clean['SEX'].map(recode1)

data_clean['S1Q7A1'] = data_clean['S1Q7A1'].map(recode1)
```

```
data_clean['S1Q14A'] = data_clean['S1Q14A'].map(recode1)

#select predictor variables and target variable as separate data sets

predvar= data_clean[['SEX','AGE','S1Q12A','S1Q14A','S1Q24LB','S1Q7A1']]

target = data_clean.S2AQ8B

# standardize predictors to have mean=0 and sd=1

predictors=predvar.copy()

from sklearn import preprocessing

predictors['SEX']=preprocessing.scale(predictors['SEX'].astype('float64'))

predictors['AGE']=preprocessing.scale(predictors['AGE'].astype('float64'))

predictors['S1Q7A1']=preprocessing.scale(predictors['S1Q7A1'].astype('float64'))

predictors['S1Q12A']=preprocessing.scale(predictors['S1Q12A'].astype('float64'))

predictors['S1Q14A']=preprocessing.scale(predictors['S1Q14A'].astype('float64'))

predictors['S1Q24LB']=preprocessing.scale(predictors['S1Q24LB'].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')


# 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')

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)
```