Kyu-DATA

categories

Sunday, January 31, 2016

Machine Learning for Data Analysis Week3 Lasso Regression

```
from pandas import Series, DataFrame
import matplotlib.pylab as plt
from sklearn.cross_validation import train_test_split
from sklearn.metrics import classification_report
import pandas as pd
import numpy as np
from sklearn import preprocessing

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

#---------------------------------------------------------------------------
# Data Cleaning
#---------------------------------------------------------------------------
# explanatory variable
predictors = data[['gender_respondent',
'interest_attention','interest_whovote2008','presapp_track','presapp_job_x','pr
'presapp_foreign_x','presapp_health_x','presapp_war_x','finance_finfam','financ
'finance_finnext_x','health_insured','health_2010hcr_x','libcpre_choose',
'divgov_splitgov','campfin_limcorp','campfin_banads','ineq_incgap_x','effic_unc
'effic_carestd','econ_ecpast_x',
'econ_ecnext_x','econ_unpast_x','ecblame_pres','ecblame_fmpr','ecblame_dem',
'tea_supp_x','gun_importance','immig_policy','fedspend_ss','fedspend_schools',
'fedspend_scitech','fedspend_crime','fedspend_welfare','fedspend_child',
'fedspend_poor','fedspend_enviro','dem_marital','dem_edugroup','dem_eduspgroup
'dem_veteran','dem_empstatus_1digitfinal','dem_raceeth','dem_parents','dem2_nur
'owngun_owngun', 'orientn_rgay', 'happ_lifesatisf']]

# target variable
targets = data['prevote_regpty']

# Convert categorical variable to numpy arrays and fill NaN values to zero.
# predictors[col] = number.fit_transform(predictors[col].replace(np.nan,'0', re
def convert(dta):
    number = preprocessing.LabelEncoder()
    for col in dta.columns:
        dta[col] = number.fit_transform(dta[col].fillna(''))
    return dta

        # Catagorizing income group function
def incgroup_prepost(row):
    if type(row) == float and np.isnan(row):
        return float('NaN')
    elif row == "$15,000-$17,499" or row == "$10,000-$12,499" or row == "$5,00
        return 1
    elif row == "$27,500-$29,999" or row == "$25,000-$27,499" or row == "$20,00
        return 2
    elif row == "$35,000-$39,999" or row == "$30,000-$34,999":
        return 3
    elif row == "$45,000-$49,999" or row == "$40,000-$44,999":
        return 4
    elif row == "$50,000-$54,999" or row == "$55,000-$59,999":
        return 5
    elif row == "$60,000-$64,999" or row == "$65,000-$69,999":
        return 6
    elif row == "$70,000-$74,999" or row == "$75,000-$79,999":
        return 7
    elif row == "$80,000-$89,999":
        return 8
```

About Me

Kyu Cho

View my complete profile

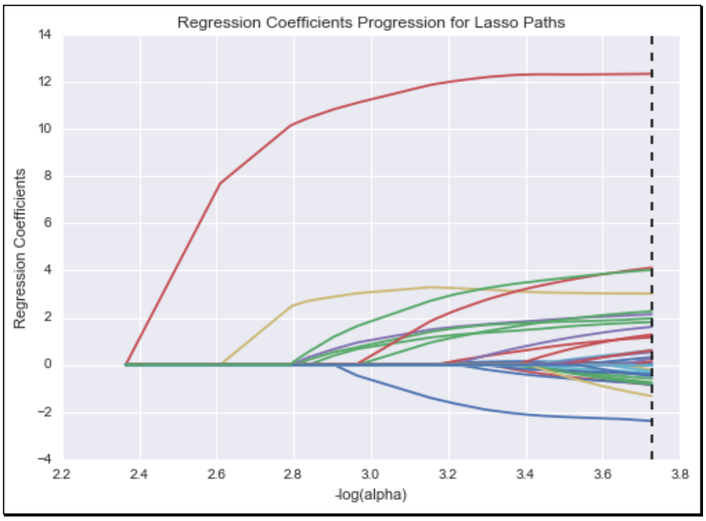
Blog Archive

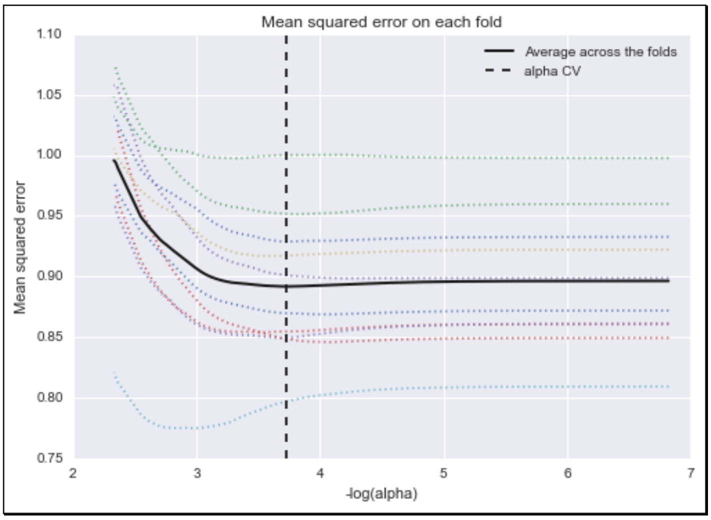
```
    elif row == "$90,000-$99,999":
        return 9
    elif row == "$100,000-$109,999":
        return 10
    elif row == "$110,000-$124,999" or row == "$125,000-$149,999":
        return 11
    elif row == "$150,000-$174,999" or row == "$175,000-$249,999":
        return 15
    elif row == "$250,000 Or More":
        return 25

# Explantory var Cleaning
predictors = convert(predictors)
predictors['incgroup_prepost'] = (data['incgroup_prepost'].apply(lambda row: i

# Response var Cleaning
number = preprocessing.LabelEncoder()
targets = number.fit_transform(targets.fillna(''))

# Spliting Data
pred_train, pred_test, tar_train, tar_test  = train_test_split(predictors, tar


#----------------------------------------------------------------------------
# Building Lasso Model
#----------------------------------------------------------------------------
# standardize predictors to have mean=0 and sd=1
import matplotlib.pylab as plt
from sklearn.linear_model import LassoLarsCV
from sklearn import preprocessing

# standardize clustering variables to have mean=0 and sd=1
predictors = predictors.copy()
def stdNscale(dta):
    for col in dta.columns:
        predictors[col] = preprocessing.scale(predictors[col].astype('float64'
    return dta
predictors = stdNscale(predictors)
targets = preprocessing.scale(targets.astype('float64'))

# split data into train and test sets
pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targe
                                                              test_size=.3, ra


# 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')
```

```python
# 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)
```

project12.py hosted with ❤ by GitHub view raw

```
Out[69]:
{'campfin_banads': 0.0,
 'campfin_limcorp': 0.0,
 'dem2_numchild': 0.0,
 'dem_edugroup': 0.0,
 'dem_eduspgroup': 0.0,
 'dem_empstatus_1digitfinal': 0.0,
 'dem_marital': 0.0093908673093667499,
 'dem_parents': 0.064314130130658162,
 'dem_raceeth': 0.030557155678705504,
 'dem_veteran': 0.0,
 'divgov_splitgov': 0.0,
 'ecblame_dem': 0.0024174774571679579,
 'ecblame_fmpr': 0.035363241515626649,
 'ecblame_pres': -0.013176501904743277,
 'econ_ecnext_x': 0.0,
 'econ_ecpast_x': 0.0,
 'econ_unpast_x': -0.0042517012173674644,
 'effic_carestd': 0.0,
 'effic_undstd': -0.011690301880860978,
 'fedspend_child': -0.0053244859938198202,
 'fedspend_crime': 0.0042714539023118291,
 'fedspend_enviro': -0.012919955395559209,
 'fedspend_poor': -0.0063130602487481121,
 'fedspend_schools': -0.0068057924372977568,
 'fedspend_scitech': 0.019934013244821611,
 'fedspend_ss': -0.037135780387225703,
 'fedspend_welfare': 0.0,
 'finance_finfam': 0.025162446494403065,
 'finance_finnext_x': 0.0096358048042361599,
 'finance_finpast_x': -0.0034004738990444745,
 'gender_respondent': 0.0050017603010246436,
 'gun_importance': 0.0,
 'happ_lifesatisf': -0.0074631875707886104,
 'health_2010hcr_x': 0.028169615424706553,
 'health_insured': 0.0,
 'immig_policy': 0.0,
 'incgroup_prepost': 0.062892261894390789,
 'ineq_incgap_x': 0.0,
 'interest_attention': -0.0090041515906069888,
 'interest_whovote2008': 0.19173136306395783,
 'libcpre_choose': -0.013095974298389566,
 'orientn_rgay': 0.0,
 'owngun_owngun': -0.020832005352398528,
 'presapp_econ_x': 0.0,
 'presapp_foreign_x': 0.0013025839386105601,
 'presapp_health_x': 0.0,
 'presapp_job_x': 0.047018777798969492,
 'presapp_track': 0.033074628032719842,
 'presapp_war_x': 0.01812625523301812,
 'tea_supp_x': 0.0083493998108720071}
```





We can see that 19 variables are removed out of 50 variables after i applied the lasso penalty. We can see that income group is positively related with respondent's political party and gun owned is negatively related with it.

training data MSE
0.877166028246
test data MSE
0.890551394001

training data R-square
0.118174140339
test data R-square
0.120284409471

We have similar MSE from training set and testing set which mean the prediction is pretty stable.

If you want to know the detail variables that is been used for this analysis, check the following link
https://d396qusza40orc.cloudfront.net/statistics%2Fproject%2Fanes1.html#incgroup_prepost

Posted by Kyu Cho at 6:51 PM

G+1 Recommend this on Google

Labels: categories

No comments:

Post a Comment

Enter your comment...

Comment as: Google Accou

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)