Module 4 | Python

November 21, 2021

1 ADS 502 Modul 4 Assignment | Python

1.0.1 Ryan S. Dunn

1.1 Data Science Using Python and R: Chapter 13 - Page 195: Questions #13, 14, 15, 16, & 17

[31]:
```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy import stats
from sklearn.metrics import confusion_matrix
```

1.1.1 13. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.

[32]:
```
clothing_test = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied␣
↪Data Mining 502/Module 4/Datasets/clothing_sales_test.csv")
#clothing_test.head()
```

[33]:
```
clothing_train = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied␣
↪Data Mining 502/Module 4/Datasets/clothing_sales_training.csv")
```

[34]:
```
#separate the training variables into predictor variables and response variables
X_train = pd.DataFrame(clothing_train[['Days','Web']])
X_train = sm.add_constant(X_train)
y_train = pd.DataFrame(clothing_train[['CC']])
```

[35]:
```
#run the logistic regression model
logreg1 = sm.Logit(y_train, X_train).fit()
logreg1.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.655955
         Iterations 5
```

[35]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.053
Dependent Variable: CC               AIC:              1909.5825
Date:               2021-11-21 09:58 BIC:              1925.4226
No. Observations:   1451             Log-Likelihood:   -951.79
Df Model:           2                LL-Null:          -1004.9
Df Residuals:       1448             LLR p-value:      8.3668e-24
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
            Coef.    Std.Err.      z      P>|z|     [0.025    0.975]
-----------------------------------------------------------------
const       0.4962     0.0887    5.5968   0.0000    0.3224    0.6699
Days       -0.0037     0.0004   -8.4491   0.0000   -0.0046   -0.0028
Web         1.2537     0.3307    3.7914   0.0001    0.6056    1.9018
=================================================================

"""
```

1.1.2 14. Are there any variables that should be removed from the model? If so, remove them and rerun the model

There should not be any variables removed from the model, as both p-values for Days and Web suggest that the co-efficients are statistically significant, and therefore relevant for the logistic regression model.

1.1.3 15. Write the descriptive form of the logistic regression model using the coefficients obtained from Question #13.

$CC = \beta_0 - \beta_1(Days) - \beta_2(Web)$

Where: $\beta_0 = 0.4962$, $\beta_1 = 0.0037$, $\beta_2 = 1.2537$

1.1.4 16. Validate the model using the test data set.

[36]:
```
#separate the test variables into predictor variables and response variables
X_test = pd.DataFrame(clothing_test[['Days','Web']])
X_test = sm.add_constant(X_test)
y_test = pd.DataFrame(clothing_test[['CC']])

#run the model using the test data set
logreg1_test = sm.Logit(y_test, X_test).fit()
logreg1_test.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.656885
```

```
         Iterations 5
```

[36]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.052
Dependent Variable: CC               AIC:              1838.7104
Date:               2021-11-21 09:58 BIC:              1854.4324
No. Observations:   1395             Log-Likelihood:   -916.36
Df Model:           2                LL-Null:          -966.40
Df Residuals:       1392             LLR p-value:      1.8534e-22
Converged:          1.0000           Scale:            1.0000
No. Iterations:     5.0000
-----------------------------------------------------------------
            Coef.    Std.Err.      z      P>|z|     [0.025    0.975]
-----------------------------------------------------------------
const       0.4634     0.0873    5.3105   0.0000    0.2924    0.6345
Days       -0.0035     0.0004   -8.2261   0.0000   -0.0043   -0.0026
Web         1.0973     0.2830    3.8780   0.0001    0.5427    1.6519
=================================================================

"""
```

1.1.5 17. Obtain the predicted values of the response variable for each record in the data set.

[37]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

#call the LogisticReggresion algorithm from sklearn
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

#obtain the predicted values from the test set
y_pred = logreg.predict(X_test)

#create the "predicted and actual" confusion matrix of the actual and predicted␣
 ↪values
conf_max = pd.DataFrame(confusion_matrix(y_test, y_pred))
conf_max
```

```
/Users/ryan_s_dunn/opt/anaconda3/lib/python3.8/site-
packages/sklearn/utils/validation.py:63: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

[37]:
	0	1
0	403	314
1	215	463

[38]:
```
#obtain the classification report for the training data
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.65      0.56      0.60       717
           1       0.60      0.68      0.64       678

    accuracy                           0.62      1395
   macro avg       0.62      0.62      0.62      1395
weighted avg       0.62      0.62      0.62      1395
```

1.2 Data Science Using Python and R: Chapter 9 - Page 138: Questions #24, 25, 26, 27, 28, 29, & 30

24. Prepare the data set for neural network modeling, including standardizing the variables.

[39]:
```
import pandas as pd
import numpy as np

#import training and test data sets
bank_test = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data␣
↪Mining 502/Module 4/Datasets/bank_marketing_test")

#import the test data set
bank_train = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data␣
↪Mining 502/Module 4/Datasets/bank_marketing_training")
```

[9]:
```
#preprocess the data for the ANN model
```

[10]:
```
#develop the ANN model
```

[ ]:
```

```

1.3 Data Science Using Python and R: Chapter 6 - Page 93: Questions #19 & 20

[40]:
```
#preprocess the data for inclusion in the random forest algorithm
import statsmodels.tools.tools as stattools
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import tree
```

[41]:
```
#import training data
training = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data␣
↪Mining 502/Module 2/Datasets/adult_ch6_training", header = 0)
```

[42]:
```
#create our y value - Income
y_train = training['Income']
```

[43]:
```
#convert categorical varialbes to dummy variables because sklearn needs␣
↪converted categorical variables
marital_status_dummy = np.array(training['Marital status'])

#save matrix and dictionary separately
(marital_cat, marital_cat_dict) = stattools.categorical(marital_status_dummy,
                                                        ␣
↪drop=True,dictnames=True)
```

```
/Users/ryan_s_dunn/opt/anaconda3/lib/python3.8/site-
packages/statsmodels/tools/tools.py:158: FutureWarning: categorical is
deprecated. Use pandas Categorical to represent categorical data and can
get_dummies to construct dummy arrays. It will be removed after release 0.13.
  warnings.warn(
```

[44]:
```
#add the dummy variables back into the X varialbes
marital_status_pd = pd.DataFrame(marital_cat)

#attach the predictor variable Cap_Gains_Losses to the data frame of dummy␣
↪variables that represent marital status
X_train= pd.concat((training[['Cap_Gains_Losses']],
                    marital_status_pd), axis=1)
```

[45]:
```
#note that the columns of X and y do not include the different values of the␣
↪marital status, so we run
#martial_cat_dict and assigne the categories to y to see the values of the␣
↪columns
X_names = ["Cap_Gains_Losses","Divorced","Married",
           "Never-married","Separated","Widowed"]
y_names = ["<=50K",">50K"]
```

1.3.1 19. Use random forests on the training data set to predict income using marital status and capital gains and losses.

[46]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import numpy as np
```

[47]:
```
#include the response variable formattted as a one dimensional array
rfy = np.ravel(y_train)
```

[48]:
```
#use the randomforest classifier to create the randome forest
rf_1 = RandomForestClassifier(n_estimators = 100, criterion ="gini").
 ↪fit(X_train,rfy)
```

[49]:
```
#obtain the predictions from the model
yrf_predict = rf_1.predict(X_train)

#return the confusion matrix for the random forest classifier
cm_yrf = pd.DataFrame(confusion_matrix(y_train, yrf_predict))
cm_yrf
```

[49]:

	0	1
0	14237	34
1	3138	1352

[50]:
```
#obtain the classification report for the training data
print(classification_report(y_train, yrf_predict))
```

```
              precision    recall  f1-score   support

       <=50K       0.82      1.00      0.90     14271
        >50K       0.98      0.30      0.46      4490

    accuracy                           0.83     18761
   macro avg       0.90      0.65      0.68     18761
weighted avg       0.86      0.83      0.79     18761
```

1.3.2 20. Use random forests using the test data set that utilizes the same target and predictor variables. Does the test data result match the training data result?

[51]:
```
#import test data
test = pd.read_csv("/Users/ryan_s_dunn/Documents/USD MS-ADS/Applied Data Mining␣
 ↪502/Module 2/Datasets/adult_ch6_test", header = 0)
```

[52]:
```
test.head(1)
```

[52]:

	Marital status	Income	Cap_Gains_Losses
0	Married	<=50K	0.0

[53]:
```
y_test = test['Income']

#convert categorical varialbes to dummy variables because sklearn needs␣
 ↪converted categorical variables
marital_status_dummy2 = np.array(test['Marital status'])
```

```
#save matrix and dictionary separately
(marital_cat2, marital_cat_dict2) = stattools.categorical(marital_status_dummy2,
                                                          ␣
 ↪drop=True,dictnames=True)

#add the dummy variables back into the X varialbes
marital_status_pd2 = pd.DataFrame(marital_cat2)

#attach the predictor variable Cap_Gains_Losses to the data frame of dummy␣
 ↪variables that represent marital status
X_test= pd.concat((test[['Cap_Gains_Losses']],
                   marital_status_pd2), axis=1)

#note that the columns of X and y do not include the different values of the␣
 ↪marital status, so we run
#martial_cat_dict and assigne the categories to y to see the values of the␣
 ↪columns
X_names2 = ["Cap_Gains_Losses","Divorced","Married",
           "Never-married","Separated","Widowed"]
y_names2 = ["<=50K",">50K"]
```

```
/Users/ryan_s_dunn/opt/anaconda3/lib/python3.8/site-
packages/statsmodels/tools/tools.py:158: FutureWarning: categorical is
deprecated. Use pandas Categorical to represent categorical data and can
get_dummies to construct dummy arrays. It will be removed after release 0.13.
  warnings.warn(
```

[72]:
```
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

#include the response variable formattted as a one dimensional array
rfy2 = np.ravel(y_test)

#use the randomforest classifier to create the randome forest
rf_2 = RandomForestClassifier(n_estimators = 100, criterion ="gini").
 ↪fit(X_test,rfy2)

#obtain the predictions from the model
yrf_predict_test = rf_2.predict(X_test)

#return the confusion matrix for the random forest classifier

cm_yrf = pd.DataFrame(confusion_matrix(y_test, yrf_predict_test))
print(' Model confusion matrix: ')
print(cm_yrf)
```

```
 Model confusion matrix:
```

```
      0    1
0  4668    6
1  1034  447
```

[71]:
```
#obtain the classification report for the training data
print(classification_report(y_test, yrf_predict_test))
```

```
              precision    recall  f1-score   support

       <=50K       0.82      1.00      0.90      4674
        >50K       0.99      0.30      0.46      1481

    accuracy                           0.83      6155
   macro avg       0.90      0.65      0.68      6155
weighted avg       0.86      0.83      0.79      6155
```

[56]:
```
#obtain the classification report for the training data
print(classification_report(y_train, yrf_predict))
```

```
              precision    recall  f1-score   support

       <=50K       0.82      1.00      0.90     14271
        >50K       0.98      0.30      0.46      4490

    accuracy                           0.83     18761
   macro avg       0.90      0.65      0.68     18761
weighted avg       0.86      0.83      0.79     18761
```

The test training result does match the test training results.