Week 2 Assignment

By: Valon Tika

Summary and Problem Definition

Companies can leverage the use of telemarketing campaigns as a route to reach out to the market and to offer numerous services. This can lead to higher sales for a firm due to penetration of the market by the way of calling to customers. Companies in the financial services industry see this as a great opportunity to sell financial products to retain higher gross margins and to have more assets to allocate to possibly retain more earnings. One way to achieve this is to target customers that are probably more likely to participate in a given program through the use of a telemarketing campaign, rather than random dialing to customers.

Research Design, Measurement and Statistical Method

For this analysis, we can use the data from previous executed marketing campaigns and observe the data that was retained within those campaigns such as job occupation, housing, education, whether they have a personal loan, and other key pieces of information about the customer. It also contains financial related information such as outstand loan amounts and average account balances.

There was around 11.5% of total respondents that said yes in this marketing campaign. Out of those 11.5%, the average balance was around $1,570 dollars. Most of the individuals that response yes was contacted a little over 2 times on average during the campaign. Very few individuals that had a balance between $5,000 to $10,000 had a 15% on average of being interested in the campaign compared to the 75% of respondents who's account balances were under $5,000.

Overview of Programming Work

The analysis of the marketing campaign data was performed in Python, a general-purpose programming language that provides many tools and packages to perform data analysis, data modeling, and predictive modeling in an efficient manner. One specific variable that seemed to show some impact to interest was the average balance, which was $1,403 that were not interested vs $1,572 for those that were. Balance, housing and loans were economic metrics deemed as the larger factors of interest. Those three metrics were used to compare 2 classification models. Naïve Bayes and Logistic Regression binary classifiers were chosen due to the robustness of these models for this scenario.

The classifiers will be different based on the model used. Performance of the models will be measured by using a standard performance metric, the ROC curve, or Receiver Operating Characteristic curve. This metric perspective of accuracy of predictions that will be relative to the TPR (true positive rate), as if we dial a customer on a false-positive, the negative outcome would be that they will reject, although they might not.

Review of Results – Recommendations

Since the ROC models can’t be compared due to the training data being heavily biased for one class, training accuracy can be a valid criterion to evaluate the model. Training accuracy on logistic regression is 88% and 87% on Gaussian Naïve Bayes, hence the logistic regression is just 1% better than Naïve Bayes. Also, logistic regression performs much better on test data, but we can’t use test data for comparison. The training accuracy is actually the mean of training + validation accuracy

Code Appendix:

```
#!/usr/bin/env python

# coding: utf-8



# In[ ]:




# Jump-Start for the Bank Marketing Study

# as described in Marketing Data Science: Modeling Techniques

# for Predictive Analytics with R and Python (Miller 2015)


# jump-start code revised by Thomas W. Milller (2018/10/07)


# Scikit Learn documentation for this assignment:

# http://scikit-learn.org/stable/auto_examples/classification/

#   plot_classifier_comparison.html

# http://scikit-learn.org/stable/modules/generated/

#   sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB.score
```

```
# http://scikit-learn.org/stable/modules/generated/
#   sklearn.linear_model.LogisticRegression.html
# http://scikit-learn.org/stable/modules/model_evaluation.html
# http://scikit-learn.org/stable/modules/generated/
#  sklearn.model_selection.KFold.html

# prepare for Python version 3x features and functions
# comment out for Python 3.x execution
# from __future__ import division, print_function
# from future_builtins import ascii, filter, hex, map, oct, zip


# In[2]:


# seed value for random number generators to obtain reproducible results
RANDOM_SEED = 1
```

```
# import base packages into the namespace for this program

import numpy as np

import pandas as pd




# In[3]:




# initial work with the smaller data set

bank = pd.read_csv('bank.csv', sep = ';')  # start with smaller data set

# examine the shape of original input data

print(bank.shape)




# In[4]:




# drop observations with missing data, if any
```

```
bank.dropna()

# examine the shape of input data after dropping missing data

print(bank.shape)




# In[5]:




# look at the list of column names, note that y is the response

list(bank.columns.values)




# In[6]:




# look at the beginning of the DataFrame

bank.head()
```

```
# In[7]:




# mapping function to convert text no/yes to integer 0/1

convert_to_binary = {'no' : 0, 'yes' : 1}




# In[8]:




# define binary variable for having credit in default

default = bank['default'].map(convert_to_binary)




# In[9]:
```

```
# define binary variable for having a mortgage or housing loan

housing = bank['housing'].map(convert_to_binary)




# In[10]:




# define binary variable for having a personal loan

loan = bank['loan'].map(convert_to_binary)


# define response variable to use in the model

response = bank['response'].map(convert_to_binary)




# In[11]:




# gather three explanatory variables and response into a numpy array
```

```
# here we use .T to obtain the transpose for the structure we want

model_data = np.array([np.array(default), np.array(housing), np.array(loan),

    np.array(response)]).T




# In[12]:




# examine the shape of model_data, which we will use in subsequent modeling

print(model_data.shape)




# In[13]:




# the rest of the program should set up the modeling methods

# and evaluation within a cross-validation design
```

```
# In[14]:

features = model_data[:,:3]
output = model_data[:,-1]

# # Initial Exploratory Analysis

# In[39]:

bank['response'].value_counts().plot(kind='bar')

# it looks as though there is a small amount of individuals that have responded to the campaign.
let's look at the %.
```

In[41]:

(response == 1).sum() / len(response * 100)

In[45]:

##Around 11.5% responded yes to the campaign. let's look at some of the demographics of ones that said yes

yesresponse = bank.loc[bank['response'] == 'yes']

yesresponse.groupby('response').mean()

In[52]:

##Average age is around 42.9 years old, average days since contact of around 15.65 and contacted around 2.26 times

##I want to look a histogram of the age, balance, day, campaign and duration to see the normality of the data

distro = yesresponse[['age', 'balance', 'day', 'campaign', 'duration']]

distro.hist(bins = 5, figsize = (18,5), layout=(1,5))

All the indiduals that had said yes seem to have low balances (below 10k), very low duration, and very low number of contacts during the campaign. I want to see the individuals below a 5 campai

In[55]:

distro2 = distro.loc[distro['campaign'] <= 5]

```
distro2.hist( bins = 5, figsize = (18,5), layout=(1,5))
```

Let's filter this even more to see ones with a balance of less than or equal to 5k and a duration of less than or equal to 1000.

In[59]:

```
distro3 = distro2.loc[distro2['balance'] <= 5000]

distro3 = distro3.loc[distro3['duration']<=1000]

distro3.hist( bins = 5, figsize = (18,5), layout=(1,5))
```

Even though there is still skewness within the data, the normality of the data seems better for both balance and duration for the same age group on average.

```
# # Logistic Regression Modelling method

# In[31]:

# importing the libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

# In[32]:
```

```
# spliting the dataset between train and test.

# training dataset is 70% and test data is 30%

X_train, X_test, y_train, y_test = train_test_split(features, output, test_size=0.30,
random_state=42)


# In[33]:


# Logistic Regression classifier using scikit learn

clf = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial')

clf.fit(X_train, y_train)


# In[34]:


# k fold cross validation where k=5

# validation set is prepared from the training data internally by scikit learn.
```

```
k=5

logistic_scores = cross_val_score(clf, X_train, y_train, cv=k)




# In[35]:





# calculate the accuracy and standard deviation in the accuracy

logistic_accuracy = logistic_scores.mean()

logistic_std = logistic_scores.std()


print("Training Accuracy on logistic regression: %0.2f (+/- %0.2f)" % (logistic_accuracy,
logistic_std * 2))




# This means that logsitic regression model is 88% accurately trained and with 0 standard
deviation on the training data.
```

```
# In[36]:

# test predictions
y_predict = clf.predict(X_test)

# In[37]:

print('Test accuracy on logistic regression: %0.2f' % clf.score(X_test, y_test))

# In[38]:

# ROC plot
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_predict, pos_label=2)
```

the warning tells us that model is too much biased but since the accuracy is good, that means training data itself is biased. there are 4000 records with no as the response and 500 records with yes as response.

In[23]:

```
print('no count: ', bank[bank['response']=='no'].count()[0])

print('yes count: ', bank[bank['response']=='yes'].count()[0])
```

Naive Bayes Classifier modelling technique

In[24]:

importing the naive bayes module

```
from sklearn.naive_bayes import GaussianNB


# In[25]:


# gaussian naive bayes model instance
gnb = GaussianNB()

# training on the training data
gnb.fit(X_train, y_train)


# In[26]:


# k fold cross validation where k=5
# validation set is prepared from the training data internally by scikit learn.
```

```
k=5

gnb_scores = cross_val_score(gnb, X_train, y_train, cv=k)


# In[27]:


# calculate the accuracy and standard deviation in the accuracy

gnb_accuracy = gnb_scores.mean()

gnb_std = gnb_scores.std()


print("Training Accuracy on gnb: %0.2f (+/- %0.2f)" % (gnb_accuracy, gnb_std * 2))


# In[28]:
```

```
# test predictions

y_predict = gnb.predict(X_test)

# In[29]:

print('Test accuracy on gnb: %0.2f' % gnb.score(X_test, y_test))

# In[30]:

# ROC plot

fpr, tpr, thresholds = metrics.roc_curve(y_test, y_predict, pos_label=2)
```

the warning tells us that model is too much biased but since the accuracy is good, that means training data itself is biased. there are 4000 records with no as the response and 500 records with yes as response.

In[157]:

```
print('no count: ', bank[bank['response']=='no'].count()[0])

print('yes count: ', bank[bank['response']=='yes'].count()[0])
```

Since ROC cant be compared as the training data is heavily biased for one class, training accuracy can be a good criteria to evaluate the model. Training accuracy on logistic regression is 88% and 87% on gaussian naive bayes. hence logistic regression is just 1% better than naive bayes. Also logistic regression performs much better on test data, but we cant use test data for comparison. training accuracy is actually the mean of training + validation accuracy

In[ ]: