Building multiple machine learning models for success of bank telemarketing using scikit-learn

Yupeng Wang, Ph.D., Data Scientist

Overview

Machine learning (ML) has been increasingly used for guiding marketing activities. In this demo, I build multiple ML models using the Python scikit-learn library to predict success of bank telemarketing, i.e. whether a customer subscribes a term deposit, using customer information and phone call events as predictor variables. These ML models achieve accuracy around 90%. Important predictor variables are analyzed and visualized.

The source code of this demo can be downloaded from https://github.com/wyp1125/Sklearn-Bank-Marketing.

Key techniques: Python, machine learning, scikit-learn, pandas, data filtering, feature scaling, categorical variables, one-hot encoding, variable importance, matplotlib.

Raw data

Raw data were downloaded from https://archive.ics.uci.edu/ml/datasets/bank+marketing. Only the "bank-additional-full.csv" file was used. The dataset contained 41,188 records and 20 input variables. The output variable was 'y': whether a client subscribed a term deposit. The 20 input variables included both numeric and categorical variables. Detailed description of the input variables can be found from the aforementioned website.

Description of the source code

Import required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
```

Specify the classifiers to be used

```
clsr_names=["Nearest Neighbors", "Linear SVM", "RBF SVM",
            "Decision Tree", "Random Forest", "Neural Net", "AdaBoost",
            "Naive Bayes"]
classifiers = [KNeighborsClassifier(3),
            SVC(kernel="linear", C=0.025), SVC(gamma=2, C=1),
            DecisionTreeClassifier(max_depth=5),
            RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
            MLPClassifier(alpha=1),
            AdaBoostClassifier(),
            GaussianNB()]
```

Import raw data and filter out records with missing values

```
df01=pd.read_csv('bank-additional/bank-additional-full.csv', sep=';',header=0)
df02=df01.dropna(axis=1, how='all')
df=df02.dropna(axis=0, how='any')
```

Derive categorical variables and perform one hot encoding

```
cols=df.dtypes
colnms=df.columns
i=0
cat_cols=[]
for eachcol in cols:
   if eachcol.name=="object":
      cat_cols.append(colnms[i])
      i+=1
df1=pd.get_dummies(df,columns=cat_cols)
```

Derive predictor and response variables, and training and testing datasets

```
x_all=df1.iloc[:,0:(m-2)]
y_all=df1['y_yes']
x_trn, x_tst, y_trn, y_tst = train_test_split(x_all, y_all, test_size=0.8, random_state=42)
```

Normalize predictor variables using the min-max scaler

```
scaler = MinMaxScaler()
scaler.fit(x_trn)
x_trn_n=scaler.transform(x_trn)
x_tst_n=scaler.transform(x_tst)
```

Fit a linear Support Vector Machine model

```
clf = classifiers[1]
model=clf.fit(x_trn_n,y_trn)
y_pred=model.predict(x_tst_n)
acc1=float((y_pred==y_tst).sum())/float(len(y_tst))
print("Linear SVM accuracy: {0:.3f}%".format(acc1))
```

Derive the top 10 import variables

```
weight=model.coef_[0]
var2wgt=pd.DataFrame(list(zip(list(df1),weight)),columns=['variable','weight'])
var2wgt_sorted=var2wgt.reindex(var2wgt.weight.abs().sort_values(ascending=False).index)
print("Top 10 weighted variables:")
print(var2wgt_sorted[0:10])
```

Make a bar plot to visualize the top 10 important variables

```
var_names=list(var2wgt_sorted['variable'][0:10])
var_imp=list(var2wgt_sorted['weight'][0:10].abs())
y_pos = np.arange(len(var_names),0,-1)
fig = plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.barh(y_pos, var_imp, align='center', alpha=0.5)
plt.yticks(y_pos, var_names)
plt.xlabel('Weight')
plt.title('Linear SVM')
plt.ylim(0,11)
```

Fit a Random Forest model

```
clf=classifiers[4]
model=clf.fit(x_trn_n,y_trn)
y_pred=model.predict(x_tst_n)
acc2=float((y_pred==y_tst).sum())/float(len(y_tst))
print("Random forest accuracy: {0:.3f}%".format(acc2))
```

Derive the top 10 import variables

```
imp=model.feature_importances_
var2imp=dict(zip(list(df1),imp))
var2imp_sorted=pd.DataFrame(columns=['variable','weight'])
for key in sorted(var2imp, key=lambda k:abs(var2imp[k]),reverse=True):
    temp=pd.DataFrame([[key,var2imp[key]]],columns=['variable','weight'])
    var2imp_sorted=var2imp_sorted.append(temp)
print("Top 10 important variables:")
print(var2imp_sorted[0:10])
```

Make a bar plot to visualize the top 10 important variables

```
var_names=list(var2imp_sorted['variable'][0:10])
var_imp=list(var2imp_sorted['weight'][0:10])
y_pos = np.arange(len(var_names),0,-1)
plt.subplot(1, 2, 2)
plt.barh(y_pos, var_imp, align='center', alpha=0.5)
plt.yticks(y_pos, var_names)
plt.xlabel('Weight')
plt.title('Random Forest')
plt.ylim(0,11)
plt.tight_layout()
```

Save the two bar plots

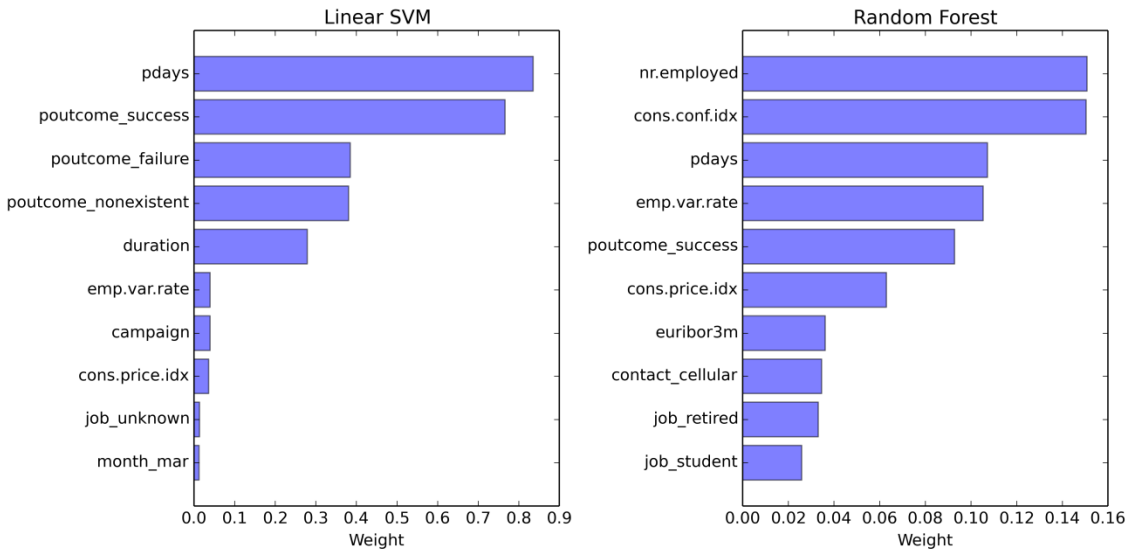
```
fig.savefig('plot.png',dpi=400
```

Compare the accuracy of different ML models

```
print("Comparing different models:")
for name, clf in zip(clsr_names, classifiers):
   model=clf.fit(x_trn_n,y_trn)
   y_pred=model.predict(x_tst_n)
   print(name+" Accuracy: {0:.3f}%".format(float((y_pred==y_tst).sum())/float(len(y_tst))))
```

Software outputs

Figure showing top 10 important variables



Screen output

```
Linear SVM accuracy: 0.898%
Top 10 weighted variables:
                variable    weight
3                  pdays -0.835169
62      poutcome_success  0.766060
60      poutcome_failure -0.385228
61  poutcome_nonexistent -0.380832
1               duration  0.278923
5           emp.var.rate -0.039887
2               campaign -0.039678
6         cons.price.idx  0.035886
21           job_unknown  0.013939
50             month_mar  0.012778
Random forest accuracy: 0.889%
Top 10 important variables:
           variable    weight
0       nr.employed  0.150851
0     cons.conf.idx  0.150417
0             pdays  0.107214
0      emp.var.rate  0.105312
0  poutcome_success  0.092824
0    cons.price.idx  0.062947
0         euribor3m  0.036129
0  contact_cellular  0.034604
0       job_retired  0.033078
0       job_student  0.025856
Comparing different models:
Nearest Neighbors Accuracy: 0.888%
Linear SVM Accuracy: 0.898%
RBF SVM Accuracy: 0.882%
Decision Tree Accuracy: 0.914%
Random Forest Accuracy: 0.888%
Neural Net Accuracy: 0.901%
AdaBoost Accuracy: 0.909%
Naive Bayes Accuracy: 0.782%
```