Faddy Sunna HW 5 - Support Vector Machine: Predict Income
PHYS 243 06/11/2019

I. **DATA:** US Gov Census. 48,842 Instances. With a 70/30 Train/Test split.

II. **Processing**: The categorical data will need to be converted into numerical real value data in order to utilize SVM. I found Sklearn Label encoder was able to convert string features into integers, but I would also need to use One hot encoder as to remove any assumed values between the integers. (EX: if USA becomes 0 and Canada becomes 1, One hot encoder will prevent the code from assuming 1 > 0 , or Canada > USA)

In the end I used Pandas.get_dummies. Which performs label encoding and one hot encoding much simpler. Shown in Figure 1 below.

```
#Import data and add column labels
data = pd.read_csv('/home/fubunutu/PycharmProjects/hw5/adult.data', header=None, index_col=False, names=['age', 'workclass', 'fnlwgt', 'education',
                                                                                        'education-num', 'marital-status', 'occupation',
                                                                                        'relationship', 'race', 'gender', 'capital-gain',
                                                                                        'capital-loss', 'hours-per-week', 'native-country',
                                                                                        'income'])




#Print features before and after one hot encoding with Pandas dummies
print('Original Labels:\n', list(data.columns), '\n')
data_dummies = pd.get_dummies(data)
print('Labels after One-Hot Encoding with Pandas dummies:\n', list(data_dummies.columns))




#Selecting all collumns for x values except income
features = data_dummies.ix[:, 'age':'native-country_ Yugoslavia']
X = features.values

#Setting y values = income > 50k, we will be predicting if income is over 50k
y = data_dummies['income_ >50K'].values
```

Fig 1: Adding column labels and one hot encoding with Pandas.dummies

Original Labels:
['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']

Labels after One-Hot Encoding with Pandas dummies:

['age', 'fnlwgt', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week', 'workclass_ ?', 'workclass_ Federal-gov', 'workclass_ Local-gov', 'workclass_ Never-worked', 'workclass_ Private', 'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc', 'workclass_ State-gov', 'workclass_ Without-pay', 'education_ 10th', 'education_ 11th', 'education_ 12th', 'education_ 1st-4th', 'education_ 5th-6th', 'education_ 7th-8th', 'education_ 9th', 'education_ Assoc-acdm', 'education_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate', 'education_ HS-grad', 'education_ Masters', 'education_ Preschool', 'education_ Prof-school', 'education_ Some-college', 'marital-status_ Divorced', 'marital-status_ Married-AF-spouse', 'marital-status_ Married-civ-spouse', 'marital-status_ Married-spouse-absent', 'marital-status_ Never-married', 'marital-status_ Separated', 'marital-status_ Widowed', 'occupation_ ?', 'occupation_ Adm-clerical', 'occupation_ Armed-Forces', 'occupation_ Craft-repair', 'occupation_ Exec-managerial', 'occupation_ Farming-fishing', 'occupation_ Handlers-cleaners', 'occupation_ Machine-op-inspct', 'occupation_ Other-service', 'occupation_ Priv-house-serv', 'occupation_ Prof-specialty', 'occupation_ Protective-serv', 'occupation_ Sales', 'occupation_ Tech-support', 'occupation_ Transport-moving', 'relationship_ Husband', 'relationship_ Not-in-family', 'relationship_ Other-relative', 'relationship_ Own-child', 'relationship_ Unmarried', 'relationship_ Wife', 'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black', 'race_ Other', 'race_ White', 'gender_ Female', 'gender_ Male', 'native-country_ ?', 'native-country_ Cambodia', 'native-country_ Canada', 'native-country_ China', 'native-country_ Columbia', 'native-country_ Cuba', 'native-country_ Dominican-Republic', 'native-country_ Ecuador', 'native-country_ El-Salvador', 'native-country_ England', 'native-country_ France', 'native-country_ Germany', 'native-country_ Greece', 'native-country_ Guatemala', 'native-country_ Haiti', 'native-country_ Holand-Netherlands', 'native-country_ Honduras', 'native-country_ Hong', 'native-country_ Hungary', 'native-country_ India', 'native-country_ Iran', 'native-country_ Ireland', 'native-country_ Italy', 'native-country_ Jamaica', 'native-country_ Japan', 'native-country_ Laos', 'native-country_ Mexico', 'native-country_ Nicaragua', 'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru', 'native-country_ Philippines', 'native-country_ Poland', 'native-country_ Portugal', 'native-country_ Puerto-Rico', 'native-country_ Scotland', 'native-country_ South', 'native-country_ Taiwan', 'native-country_ Thailand', 'native-country_ Trinadad&Tobago', 'native-country_ United-States', 'native-country_ Vietnam', 'native-country_ Yugoslavia', 'income_ <= 50k', 'income_ >50k']

```
#split into train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.30)

#create random forest classfier var
RF = RandomForestClassifier(n_jobs=-1)

#create SVM classifier var
svmmodel = svm.SVC(gamma='scale', degree=2)
#fit data to models
svmmodel.fit(X_train, y_train)
RF.fit(X_train, y_train)

#print accuracy scores for SVM and Random forest
print('Random forest score on the test set: {:.2f}'.format(RF.score(X_test, y_test)))
print('SVM score on the test set: {:.2f}'.format(svmmodel.score(X_test, y_test)))


y_predSVM = svmmodel.predict(X_test)
y_predRF = RF.predict(X_test)
```

Fig 2: Train Test Split, and simple model initiation

III. **Results**

In predicting income > 50k, Random Forest Accuracy = 85%, while SVM = 80% . Which should be expected because Random Forest is an ensemble method. An ensemble of SVM would probably out perform Random Forest.

Both models also had a high false positive and negative rate. As shown in the confusion matrices in Figure 4 and 5. This is most likely due to the data being heavily imbalanced, with many more instances for income < 50k than > 50k.

```
Random forest score on the test set: 0.85
SVM score on the test set: 0.80
Normalized confusion matrix
[[0.99830357 0.00169643]
 [0.84448306 0.15551694]]
Normalized confusion matrix
[[0.93258929 0.06741071]
 [0.42050391 0.57949609]]
```

Figure 3: Code output

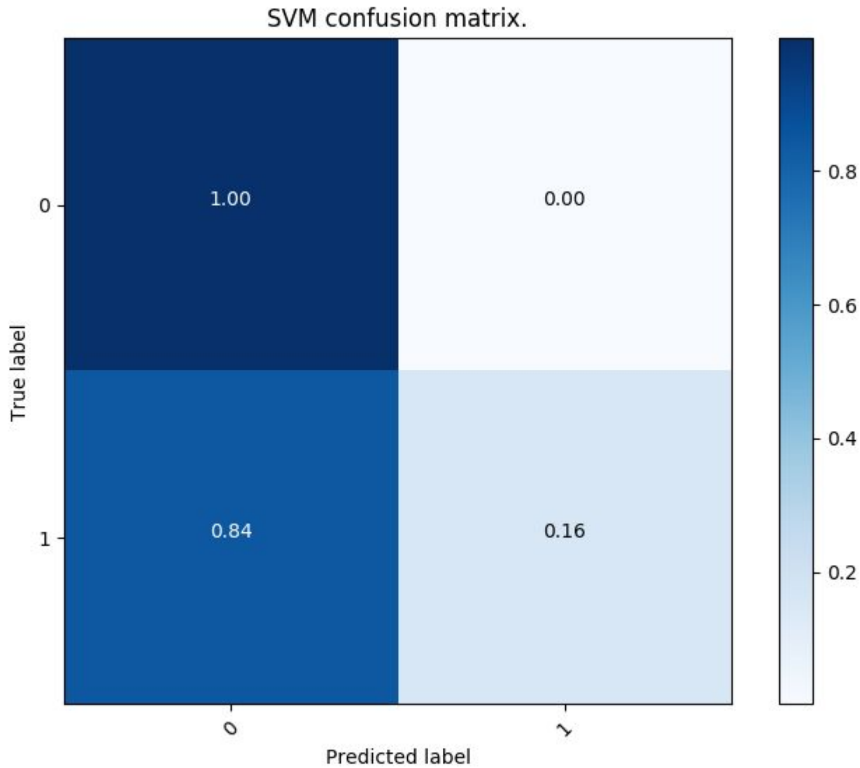

Figure 4: Normalized Support Vector Machine confusion matrix

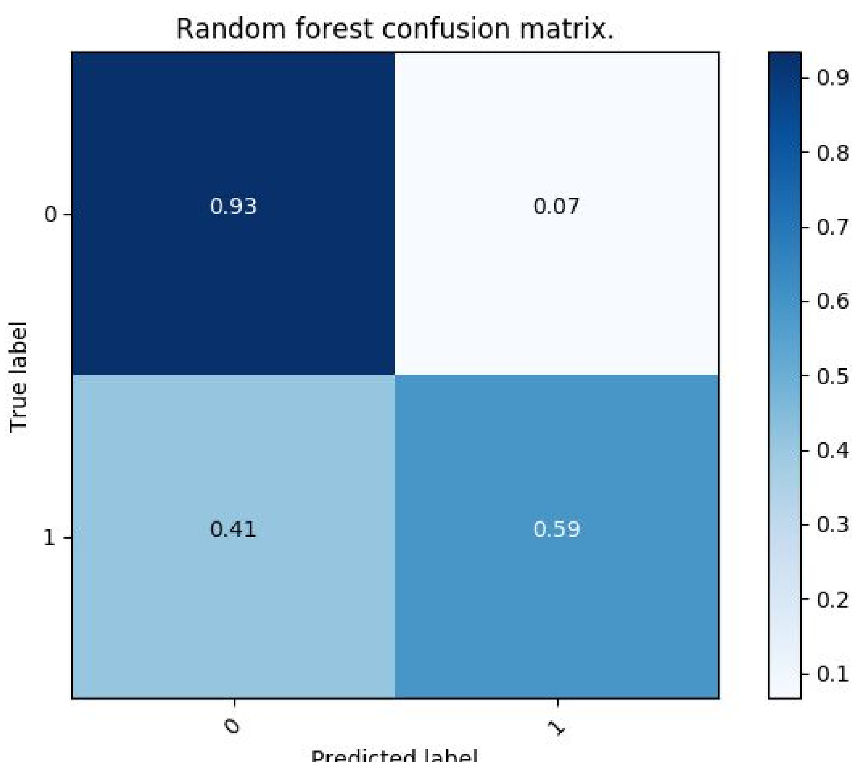

Figure 5: Normalized Random Forest confusion matrix