<u>Problem 01</u>: Classification using SVM with Python and R

Training Dataset:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes

Test Dataset:

age	income	student	credit_rating	buys_computer
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Python Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix


# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-10-tuples.csv')
test = pd.read_csv('../Datasets/test-data-4-tuples.csv')

# LabelEncoder to convert categorical to numeric value.
number = LabelEncoder()

# Convert categorical values to numeric.
for i in train:
    train[i] = number.fit_transform(train[i].astype('str'))

# Split input and output columns; x = input columns, y = output columns.
x_train = train.iloc[:, :-1]
y_train = train.iloc[:, -1]
# Do the same for test dataset.
for i in test:
    test[i] = number.fit_transform(test[i].astype('str'))
```

```
x_test = test.iloc[:, :-1]
y_test = test.iloc[:, -1]

# Build and train SVM Classifier
SVM_Classifier = SVC(kernel='linear')
SVM_Classifier.fit(x_train, y_train)

# Predict on test-data
predicted = SVM_Classifier.predict(x_test)

# Print classification report
print(classification_report(y_test, predicted))

# Build confusion matrix
cfm = confusion_matrix(y_test, predicted)
# Calc accuracy
acc = accuracy_score(y_test, predicted)

# Print acc and cfm
print('Accuracy:', acc)
print('Prediction  no  yes')
print('        no  {}   {}'.format(cfm[0][0], cfm[0][1]))
print('       yes  {}   {}'.format(cfm[1][0], cfm[1][1]))
```

Output:

```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       1.00      0.67      0.80         3

   micro avg       0.75      0.75      0.75         4
   macro avg       0.75      0.83      0.73         4
weighted avg       0.88      0.75      0.77         4

Accuracy: 0.75
Prediction  no  yes
        no  1   0
       yes  1   2
```

R Code:

```
library(RWeka)
library(caret)

train <- read.csv(file.choose())
test <- read.csv(file.choose())
model <- train(buys_computer~., method='svmLinear', data = train)
prediction <- predict(model, test)
cfMatrix <- confusionMatrix(data=prediction, test$buys_computer)
cfMatrix
```

Output:

```
> cfMatrix
Confusion Matrix and Statistics

          Reference
Prediction no yes
       no   1   2
       yes  0   1

               Accuracy : 0.5
                 95% CI : (0.0676, 0.9324)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : 0.9492

                  Kappa : 0.2
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.3333
         Pos Pred Value : 0.3333
         Neg Pred Value : 1.0000
             Prevalence : 0.2500
         Detection Rate : 0.2500
   Detection Prevalence : 0.7500
      Balanced Accuracy : 0.6667

       'Positive' Class : no
```

Problem 02: Testing Class With Unknown Data using SVM with Python and R

Training Dataset:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Test Dataset:

age	income	student	credit_rating
youth	medium	yes	fair

Python Code:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC

# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-14-tuples.csv')
test = pd.read_csv('../Datasets/unknown-classed-tuple.csv')

# LabelEncoder to convert categorical to numeric value.
number = LabelEncoder()

# Convert categorical values to numeric.
for i in train:
    train[i] = number.fit_transform(train[i].astype('str'))

# Split input and output columns; x = input columns, y = output columns.
x_train = train.iloc[:, :-1]
y_train = train.iloc[:, -1]

# Do the same for test dataset.
for i in test:
    test[i] = number.fit_transform(test[i].astype('str'))
```

```
x_test = test.iloc[:]

# Build and train SVM Classifier
SVM_Classifier = SVC(kernel='linear')
SVM_Classifier.fit(x_train, y_train)

# Do a prediction on unknown dataset.
predictions = SVM_Classifier.predict(x_test)

# Print the predicted results.
for i in predictions:
    print('Prediction: yes') if i == 1 else print('Prediction: no')
```

Output:

```
Prediction: yes
```

R Code:

```
library(RWeka)
library(caret)

data <- read.csv(file.choose())
test <- read.csv(file.choose())
classification <- train(buys_computer~., method="svmLinear", data = data)
prediction <- predict(classification, test)
prediction
```

Output:

```
> prediction
[1] yes
Levels: no yes
```

<u>Problem 03</u>: Finding Accuracy When Cross Validate, k = 2 Using SVM with Python and R

Training Dataset:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Python Code:

```
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-14-tuples.csv')

# LabelEncoder to convert categorical to numeric value.
number = LabelEncoder()

# Convert categorical values to numeric.
for i in train:
    train[i] = number.fit_transform(train[i].astype('str'))

# Create SVM Model
SVM_Classifier = SVC(kernel='linear')

# Create kFolds
kf = KFold(n_splits=2).split(train)

total = 0  # sum of the accuracies.
length = 0  # length of the kFolds

# Now loop for all the folds and predict, then sum the accuracies.
for train_indices, test_indices in kf:
    tmp_train = train.iloc[train_indices]
```

```
    tmp_test = train.iloc[test_indices]
    x_train = tmp_train.iloc[:, :-1]  # Upto last column exclusively.
    y_train = tmp_train.iloc[:, -1]  # Only the last column, i.e.
buys_computer.
    x_test = tmp_test.iloc[:, :-1]
    y_test = tmp_test.iloc[:, -1]

    #  Train/Feed the dataset to the model.
    SVM_Classifier.fit(x_train, y_train)

    # Make prediction on the test set.
    predicted = SVM_Classifier.predict(x_test)

    acc = accuracy_score(y_test, predicted)
    print('Accuracy for fold {} : {}'.format(length, acc))
    # Sum the accuracy.
    total += acc
    # Keep track the length of the kFolds.
    length += 1

# Now take the average of the accuracies.
print('\tAverage Accuracy:', total / length)
```

Output:

```
Accuracy for fold 0 : 0.42857142857142855
Accuracy for fold 1 : 0.5714285714285714
        Average Accuracy: 0.5
```

R Code:

```
library(RWeka)
library(caret)

data <- read.csv(file.choose())
kfolds <- createFolds(data$buys_computer, k = 2)
sum = 0
for(i in kfolds){
  trainData <- data[-i,]
  test <- data[i,]
  model <- train(buys_computer~., method='svmLinear', data = trainData)
  prediction <- predict(model, test)
  cfMatrix <- confusionMatrix(data = prediction, test$buys_computer)
  sum <- sum + cfMatrix$overall[1]
}
accuracy <- sum/length(kfolds)
accuracy
```

Output:

```
> accuracy
Accuracy
     0.5
```