Problem 01: Classification using Neural Network with Python

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	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 confusion_matrix, accuracy_score
import numpy
from keras.models import Sequential
from keras.layers import Dense

# Set a random seed form weight matrices.
numpy.random.seed(2)

# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-14-tupples.csv')
test = pd.read_csv('../Datasets/test-data-4-tupples.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]

# Create a sequential ANN model.
model = Sequential()
# Add first layer; neurons = 10, inputs = 4.
model.add(Dense(10, input_dim=4, activation='relu'))

# Add second layer; neurons = 4.
model.add(Dense(4, activation='relu'))

# Add output layer; 1 neron for output 0 or 1.
model.add(Dense(1, activation='sigmoid'))
# Compile this model.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Now train-up the model, iterations = 150, batch = 10.
model.fit(x_train, y_train, epochs=150, batch_size=10)

# Do a prediction on 4-tuple test dataset.
predictions = model.predict(x_test)
predicted = [int(round(x[0])) for x in predictions]

# 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:

```
Accuracy: 1.0
Prediction  no  yes
        no  1   0
       yes  0   3
```

Problem 02: Testing Class With Unknown Data using Neural Network with Python

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
import numpy
from keras.models import Sequential
from keras.layers import Dense

# Set a random seed form weight matrices.
numpy.random.seed(7)

# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-14-tupples.csv')
test = pd.read_csv('../Datasets/unknown-classed-tupple.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[:]

# Create a sequential ANN model.
model = Sequential()
# Add first layer; neurons = 10, inputs = 4.
model.add(Dense(10, input_dim=4, activation='relu'))

# Add second layer; neurons = 4.
model.add(Dense(4, activation='relu'))

# Add output layer; 1 neron for output 0 or 1.
model.add(Dense(1, activation='sigmoid'))
# Compile this model.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Now train-up the model, iterations = 150, batch = 10.
model.fit(x_train, y_train, epochs=150, batch_size=10)

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

# Result of the predictions.
outputs = [int(round(x[0])) for x in predictions]

# Print the predicted results.
for i in outputs:
    print('Prediction: Yes') if i == 1 else print('Prediction: No')
```

Output:

```
Prediction: Yes
```

Problem 03: Finding Accuracy When Cross Validate, k = 2 Using Neural Network with Python

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
import numpy
from keras.models import Sequential
from keras.layers import Dense

# Set a random seed form weight matrices.
numpy.random.seed(7)

# Read training and test dataset.
train = pd.read_csv('../Datasets/training-data-14-tupples.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 a sequential ANN model.
model = Sequential()
```

```
# Add first layer; neurons = 10, inputs = 4.
model.add(Dense(10, input_dim=4, activation='relu'))

# Add second layer; neurons = 4.
model.add(Dense(4, activation='relu'))

# Add output layer; 1 neron for output 0 or 1.
model.add(Dense(1, activation='sigmoid'))
# Compile this model.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
# 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.
    model.fit(x_train, y_train, epochs=150, batch_size=10)

    # Make prediction on the test set.
    predicted = model.predict(x_test)
    # Sum the accuracy.
    total += accuracy_score(y_test, [int(round(x[0])) for x in
predicted])
    # Keep track the length of the kFolds.
    length += 1

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

Output:

```
Accuracy: 0.6428571428571428
```