**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