CS5811 Project Source Codes Instruction

Anjia Wang (anjiaw@mtu.edu)

1 Naive Bayesian Network in Python 3

Put Python code file with dataset file "car.data" in the same folder and execute the codes in Python 3 environment by "python3 Used_Car_Naive_Bayes_Python.py". It will run several times (set by *cv* variable) and output corresponding accuracy. Then an average accuracy will be calculated. It is written by Anjia Wang and runs on Linux or macOS.

```
def preprocess(data, testsize): # testsize is % of testing
    set
    df = pandas.read_csv(data, header=None) # generate
       training and testing set
    training_set, testing_set = train_test_split(df,
       test_size = testsize)
    # generate training data and label set
    trainX = training_set.iloc[:, 0:6].values.tolist()
    trainY = training_set.iloc[:, 6].values.tolist()
    # generate testing data and label set
    testX = testing_set.iloc[:, 0:6].values.tolist()
    testY = testing_set.iloc[:, 6].values.tolist()
    return trainX, trainY, testX, testY
```

```
def term_prob_matrix(trainX, trainY):
        # calculate the term frequency in varied classes.
    label_values = numpy.zeros((4, 21))
    for i in range(0, len(trainX)):
        if trainY[i] == 'unacc':
            cl = 0
        elif trainY[i] == 'acc':
            cl = 1
        elif trainY[i] == 'good':
            cl = 2
        elif trainY[i] == 'vgood':
            cl = 3
        #check first attribute 'buying'
        if trainX[i][0] == 'vhigh':
            label_values[cl][0] += 1
        elif trainX[i][0] == 'high':
            label_values[cl][1] += 1
        elif trainX[i][0] == 'med':
            label_values[cl][2] += 1
        elif trainX[i][0] == 'low':
            label_values[cl][3] += 1
        #check second attribute 'maint'
        if trainX[i][1] == 'vhigh':
            label_values[cl][4] += 1
        elif trainX[i][1] == 'high':
            label_values[cl][5] += 1
        elif trainX[i][1] == 'med':
            label_values[cl][6] += 1
        elif trainX[i][1] == 'low':
            label_values[cl][7] += 1
        #check third attribute 'doors'
        if trainX[i][2] == '2':
            label_values[cl][8] += 1
        elif trainX[i][2] == '3':
            label_values[cl][9] += 1
        elif trainX[i][2] == '4':
            label_values[cl][10] += 1
        elif trainX[i][2] == '5more':
            label_values[cl][11] += 1
        #check fourth attribute 'persons'
        if trainX[i][3] == '2':
            label_values[cl][12] += 1
        elif trainX[i][3] == '4':
            label_values[cl][13] += 1
        elif trainX[i][3] == 'more':
            label_values[cl][14] += 1
        #check fifth attribute 'lug_boot'
        if trainX[i][4] == 'small':
            label_values[cl][15] += 1
        elif trainX[i][4] == 'med':
            label_values[cl][16] += 1
```

```python
        elif trainX[i][4] == 'big':
            label_values[cl][17] += 1
        #check sixth attribute 'safety'
        if trainX[i][5] == 'low':
            label_values[cl][18] += 1
        elif trainX[i][5] == 'med':
            label_values[cl][19] += 1
        elif trainX[i][5] == 'high':
            label_values[cl][20] += 1
    delta = 0.06
    i = 0
    for row in label_values:
        label_sum = sum(row)
        j = 0
        for v in row:
            label_values[i][j] = (1-delta)*v/label_sum +
               delta/21
            j += 1
        i += 1
    return label_values
```

```python
def calc_prior_prob(trainY):
    # calculate prior probability
    prior_prob = numpy.zeros(4)
    for i in range(0, len(trainY)):
        if trainY[i] == 'unacc':
            prior_prob[0] += 1
        elif trainY[i] == 'acc':
            prior_prob[1] += 1
        elif trainY[i] == 'good':
            prior_prob[2] += 1
        elif trainY[i] == 'vgood':
            prior_prob[3] += 1
    prior_prob = prior_prob/sum(prior_prob)
    return prior_prob
```

```python
def predict(testX, term_matrix, prior_prob):
        # prediction on testing data set.
    test_results = numpy.zeros((len(testX),4))
    for i in range(0,len(testX)):
        for j in range(0,4):
            test_results[i][j] += log(prior_prob[j])
            #check first attribute 'buying'
            if testX[i][0] == 'vhigh':
                test_results[i][j] += log(term_matrix[j
                    ][0])
            elif testX[i][0] == 'high':
                test_results[i][j] += log(term_matrix[j
                    ][1])
            elif testX[i][0] == 'med':
                test_results[i][j] += log(term_matrix[j
                    ][2])
            elif testX[i][0] == 'low':
                test_results[i][j] += log(term_matrix[j
                    ][3])
            #check second attribute 'maint'
            if testX[i][1] == 'vhigh':
                test_results[i][j] += log(term_matrix[j
                    ][4])
            elif testX[i][1] == 'high':
                test_results[i][j] += log(term_matrix[j
                    ][5])
            elif testX[i][1] == 'med':
                test_results[i][j] += log(term_matrix[j
                    ][6])
            elif testX[i][1] == 'low':
                test_results[i][j] += log(term_matrix[j
                    ][7])
            #check third attribute 'doors'
            if testX[i][2] == '2':
                test_results[i][j] += log(term_matrix[j
                    ][8])
            elif testX[i][2] == '3':
                test_results[i][j] += log(term_matrix[j
                    ][9])
            elif testX[i][2] == '4':
                test_results[i][j] += log(term_matrix[j
                    ][10])
            elif testX[i][2] == '5more':
                test_results[i][j] += log(term_matrix[j
                    ][11])
            #check fourth attribute 'persons'
            if testX[i][3] == '2':
                test_results[i][j] += log(term_matrix[j
                    ][12])
            elif testX[i][3] == '4':
                test_results[i][j] += log(term_matrix[j
                    ][13])
            elif testX[i][3] == 'more':
                test_results[i][j] += log(term_matrix[j
                    ][14])
            #check fifth attribute 'lug_boot'
            if testX[i][4] == 'small':
                test_results[i][j] += log(term_matrix[j
                    ][15])
            elif testX[i][4] == 'med':
                test_results[i][j] += log(term_matrix[j
                    ][16])
            elif testX[i][4] == 'big':
                test_results[i][j] += log(term_matrix[j
                    ][17])
            #check sixth attribute 'safety'
            if testX[i][5] == 'low':
                test_results[i][j] += log(term_matrix[j
                    ][18])
            elif testX[i][5] == 'med':
                test_results[i][j] += log(term_matrix[j
                    ][19])
            elif testX[i][5] == 'high':
                test_results[i][j] += log(term_matrix[j
                    ][20])
    pred = []
    for row in test_results:
        if numpy.argmax(row) == 0:
            pred.append('unacc')
        elif numpy.argmax(row) == 1:
            pred.append('acc')
        elif numpy.argmax(row) == 2:
            pred.append('good')
        elif numpy.argmax(row) == 3:
            pred.append('vgood')
    return pred
```

```
# main function
if __name__ == '__main__':
    cv = 10
    t_err = 0
    for k in range(0, cv):
        trainX, trainY, testX, testY = preprocess('car.
           data', 0.1)
        term_matrix = term_prob_matrix(trainX, trainY)
        prior_prob = calc_prior_prob(trainY)
        prediction = predict(testX, term_matrix,
           prior_prob)
        err = 0
        for i in range(0, len(testY)):
            if prediction[i] != testY[i]:
                err += 1
        print(1-err/len(testY))
        t_err += 1-err/len(testY)
    print(t_err/cv)
```