Qianyou Wang qianyou_wang@ucsb.edu 3494028

Report

1. Code structure

Data preprocessing function:

Used to read the data, and the corresponding format conversion

```
def trans_data(file):
    data = np.loadtxt('training.txt', dtype=np.str_, encoding='utf-8')
    column = ['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir', 'WindGustSpeed'
              'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am'
              'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday', 'RainTomorrow']
    df = pd.DataFrame(data=data, columns=column)
    df = df.replace({',': ''}, regex=True)
    column1 = ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am',
               'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
               'Temp9am', 'Temp3pm']
    for i in column1:
        df[i] = df[i].astype('float64')
```

Classification function:

To classify several categories for the data sets

```
def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated
```

Mean and variance functions:

Calculate the mean and variance of features

```
def mean(arr):
    return sum(arr)/float(len(arr))

def stdev(arr):
    avg=mean(arr)
    variance=sum([pow(x-avg,2) for x in arr])/float(len(arr)-1)
    return np.sqrt(variance)
```

Functions of categories and features:

Calculate the mean and variance values for the features corresponding to each category

```
def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for attribute in zip(*dataset)]
    del summaries[-1]
    return summaries


def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries
```

Probability function:

Calculate the probability corresponding to each feature value, and the predicted result probability of an input data

```
def calculateProbability(x, mean, stdev):
    exponent = np.exp(-(np.power(x-mean,2)/(2*np.power(stdev,2))))
    return (1 / (np.sqrt(2*np.pi) * stdev)) * exponent


def calculateClassProbabilities(summaries, inputVector):
    probabilities = {}
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= calculateProbability(x, mean, stdev)
    return probabilities
```

```
def predict(summaries, inputVector):
    probabilities = calculateClassProbabilities(summaries, inputVector)
    bestLabel, bestProb = None, -1
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel


def getPredictions(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions
```

Calculate the maximum value of the probability in the feature as the predicted value of the input data

2. Preprocessing:

Use numpy to read the txt file, however, every data contains a comma and is not a float data type.

Therefore, first turn these data into the form of a dataframe:

nshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Te
5.7,	ENE,	39.0,	ENE,	NE,	...	80.0,	71.0,	1022.8,	1021.2,	7.0,	4.0,	18.8,	
11.1,	SE,	35.0,	S,	ESE,	...	58.0,	43.0,	1017.2,	1012.9,	1.0,	1.0,	25.0,	
8.9,	N,	48.0,	NE,	NNE,	...	29.0,	18.0,	1014.9,	1012.2,	7.0,	6.0,	28.7,	
0.5,	N,	52.0,	N,	NNW,	...	69.0,	36.0,	1014.8,	1009.9,	1.0,	5.0,	12.8,	
10.9,	NE,	28.0,	ENE,	NNE,	...	80.0,	55.0,	1012.8,	1009.3,	3.0,	3.0,	27.6,	
...	...	...	...	...	...	...	...	...	...	...	...	...	
6.1,	N,	30.0,	NNE,	NNW,	...	68.0,	33.0,	1022.6,	1020.1,	6.0,	6.0,	14.1,	
10.9,	NE,	52.0,	E,	NE,	...	59.0,	51.0,	1014.7,	1011.4,	1.0,	1.0,	25.8,	
0.2,	WNW,	56.0,	NNW,	NNW,	...	61.0,	66.0,	1013.8,	1008.4,	8.0,	8.0,	10.3,	
5.0,	NE,	39.0,	ENE,	NNE,	...	89.0,	63.0,	1012.6,	1010.0,	7.0,	6.0,	19.8,	
11.3,	NNE,	30.0,	N,	S,	...	49.0,	26.0,	1021.4,	1019.2,	1.0,	2.0,	23.2,	

Then use the function in the dataframe to remove the comma and turn the corresponding feature into a float type.

3. Build the model

(1) Obtain the types of data
(2) Obtain the data corresponding to each feature
(3) Calculate the mean and variance of the data in each to calculate the probability of each feature
(4) Forecast data, select the value with the highest probability in the data of each feature as the predicted probability

4. Results

```
Accuracy: 77.31958762886599

Process finished with exit code 0
```

The training set is 80%, the test set is 20%, and the accuracy is 77%

5. Challenge

The problem is that the data structure is not standardized and contains illegal characters. Solution: use dataframe function for data preprocessing

6. Weaknesses

The model only considers the probability distribution of continuous variables, not the change of discrete features. Therefore, subsequent discrete features and continuous features can be combined, which can improve accuracy.