Machine Learning Autum-22

Assignment 1

Report

Submitted by:

Aftab Hussain(22CS60R54)

Ritik Vinod Thool(22CS60R41)

Question 2

Encoding of class attribute

The only categorical data in the dataset is the Class Attribute which is taking one of the two values ‘Abnormal’ and ‘Normal’. We have encoded ‘Abnormal’ to 0 and ‘Normal’ to 1.

The dataset is then split into 70% training and 30% testing dataset by using *train_test_split* method of *sklearn.model_selection.*

Detecting outliers (Feature selection)

By calculating mean and standard deviation of the distribution of feature values. We check for the feature values that go beyond ($2 \times \mu + 5 \times \sigma$) and if more than half of the feature values are outlier, we drop that feature.

Normalize the training set and testing set

We have implemented maxmin scalar for the normalization of the training dataset and use the training normalization parameter to normalize test data.

For every feature,

X_train_scaled = (X_train - X_train_min)/(X_train_max - X_train_min)

X_test_scaled = (X_test - X_train_min)/(X_train_max - X_train_min)

Training (Gaussian Naive bayes)

For each feature, we assume that the feature follows the Gaussian Normal distribution, so we calculate mean and standard deviation which will later be use in calculating likelihood probability in predicting the class label.

We also calculate the prior probability for both class attribute.

Testing

With respect to each instance in the testing dataset, we predict the class and compare to the ground truth (the corresponding label in testing dataset).
In predicting, instead of directly multiplying the likelihood probability, we are adding up the log of the probability so that the python will not perform round off of double values wich lead to error.

```
def predict(instance, prior_prob, gpdf_0, gpdf_1, alpha):
    post_0 = 1;

    for feature in instance:
        post_0 += math.log(normpdf(instance[feature], gpdf_0[feature][0], gpdf_0[feature][1], alpha))
    post_0 += math.log(prior_prob[0])


    post_1 = 1;

    for feature in instance:
        post_1 += math.log(normpdf(instance[feature], gpdf_1[feature][0], gpdf_1[feature][1], alpha))
    post_1 += math.log(prior_prob[1])

    if post_0 > post_1:
        return 0
    return 1
```

Log is a monotone function hence it will preserve the order and at the end we only care about the order.

5-fold cross validation

The whole dataset is divided into 5 parts, one part is used for testing and others for training, and this goes on in cycle such that every part is used as testing set exactly once.

We implemented this by slicing the dataset at appropriate index each time.
For each fold,
The training and testing set are normalized as discussed above and then model is trained on the training set and tested on the testing set. The approach of testing and training is discussed above.

We got five accuracy corresponding to each fold, The final accuracy is the mean of accuracy of each fold.

```
Accuracy for all five fold
[0.7096774193548387, 0.7741935483870968, 0.6612903225806451, 0.8225806451612904, 0.6451612903225806]
Average accuracy =  0.7225806451612903
```

Training with Laplace correction

While calculating the Gaussian normal distribution value of any feature at a test datapoint, we add a correction of alpha which will solve the zero probability problem.

```
# adding alpha to the probabilty for laplace correction
def normpdf(x, mean, sd, alpha):
    var = float(sd)**2
    denom = (2*math.pi*var)**.5
    num = math.exp(-(float(x)-float(mean))**2/(2*var))
    return (num/denom) + alpha
```

The model is trained and tested on the dataset that was split into a 70-30 training testing set and normalized thereafter using the same training and testing approach discussed above.

The accuracy is printed on the console.

```
Accuracy for 70-30 split with laplace correction
0.7096774193548387
```