Design of Embedded and Intelligent Systems, Lab 1

Group 1, Harald Lilja, Anton Olsson, Simon Brunauer

September 2019

1 Introduction

This is a short report for the first lab in the course Design of Embedded and Intelligent Systems, conducted in the autumn of 2019 at Halmstad university. The objective of the lab was to do two different tasks. The first one was to do some kind of pattern recognition based on some dataset that the student found interesting and preferrably related to the upcoming project Cyborg-terraforming. The second task was to do some kind of statistical evaluation on some dataset. The students chose to use the same dataset for this task.

2 Dataset

For the tasks at hand the students chose to use the Sonar dataset [1]. This dataset consists of two files. One consisting of 111 patterns of sonar signals bounced against a metal cylinder. The other one consists of 96 patterns of sonar signals bounced against rocks. This means that this dataset is good for classification problems where the researcher wants to differentiate between rocks and metal objects. The relation to the students project in cyborg-terraforming is that it could be further extended to classify between different types of rock formations which could prove useful in a terraforming situation.

3 Pattern recognition

The students chose to work with existing libraries, in this case scikit-learn [2]. This is due to the students having prior knowledge with pattern recognition from courses such as Artificial intelligence, Data mining and Learning systems, all conducted at Halmstad University. Since this was a smaller task not supposed to take more than two hours the students chose to try out two different classification algorithms on this dataset, the k-nearest neighbors and the support vector classifier, both are part of the scikit-learn library. This was due to them being easy to implement and to them working well with smaller datasets.

The best result was found when the parameters for the kNN-classifier was left untouched except for the amount of neighbors which was set to k = 3. This produced a cross-validation score of 79.3 %. For the SVM the best result was found when the cost-parameter was set to 0.9 and that produced a result of 76.4%.

4 Statistical evaluation

The statistical evaluation chosen for this problem was to conduct a two-sample t-test [3]. This was so that the students could test whether or not there was any average significant difference between the two groups, metal objects and rocks, this was conducted over all features for each group. The null-hypothesis here being that there in no significant difference between the two groups. The chosen alpha-value was set to 5% as that is standard. After applying the two-sample t-test on the features of the dataset, 35 of 60 features proved significantly different which means we can discard the null-hypothesis.

```
In [2]: # Split data
array = data.values
X = array[:,0:-1].astype(float)
Y = array[:,-1]
validation_size = 0.2
seed = 7
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y, test_size=validation_size, random_state=seed)
```

```
In [38]: # Classification
SVM = LinearSVC(C=0.90)
KNN = KNeighborsClassifier(3)
models = []
models.append(SVM)
models.append(KNN)
model_names = []
model_names.append('SVM')
model_names.append('KNN')

num_folds = 10
seed = 7
kfold = KFold(n_splits=num_folds, random_state=seed)
results = []
for index,model in enumerate(models):
    result = (cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy'))
    print(model_names[index] + ': ' + str(result.mean()))

SVM: 0.7639705882352941
KNN: 0.7933823529411764
```

```
In [7]: SVM.fit(X_train, Y_train)
SVM.predict(X_validation)

Out[7]: array(['M', 'M', 'R', 'M', 'M', 'M', 'M', 'R', 'R', 'R', 'M', 'R', 'M',
       'R', 'M', 'M', 'R', 'M', 'R', 'M', 'M', 'R', 'M', 'M', 'M', 'R',
       'M', 'R', 'M', 'R', 'R', 'M', 'M', 'R', 'M', 'M', 'R', 'R', 'M',
       'M', 'R', 'M'], dtype=object)
```

```
In [8]: Y_validation

Out[8]: array(['M', 'M', 'R', 'R', 'M', 'R', 'M', 'M', 'R', 'R', 'R', 'M', 'M',
       'R', 'M', 'M', 'R', 'M', 'M', 'R', 'M', 'R', 'M', 'M', 'M', 'R',
       'R', 'M', 'M', 'M', 'R', 'M', 'M', 'R', 'M', 'M', 'R', 'M', 'M',
       'M', 'R', 'M'], dtype=object)
```

Figure 1: Code used in the classification part

```
In [121]: rejected = 0
significant_features = []
for i in range(60):
    rock_feature = rocks[:,i]
    metal_feature = metal[:,i]
    data.mean()
    rock_feature_mean = np.mean(rock_feature)
    metal_feature_mean = np.mean(metal_feature)
    print("rocks mean value:",rock_feature_mean)
    print("metal mean value:",metal_feature_mean)
    rocks_std = np.std(rock_feature)
    metal_std = np.std(metal_feature)
    print("rocks std value:", rocks_std)
    print("metal std value:", metal_std)
    ttest,pval = ttest_ind(rock_feature, metal_feature)
    print("p-value",pval)
    if pval <0.05: #alpha = 5%(standard)
      print("we reject null hypothesis")
      rejected+=1
      significant_features.append({"index":i, "rock mean": rock_feature_mean, "rock std value": rocks_std, "metal mean":
    else:
      print("we accept null hypothesis")
print(str(rejected) + " of 60 features are significantly different")

rocks mean value: 0.006659375000000002
metal mean value: 0.009060360360360036
rocks std value: 0.004837776222195654
metal std value: 0.007425925844291416
p-value 0.007593714842870551
we reject null hypothesis
rocks mean value: 0.007058333333333334
metal mean value: 0.008695495495495493
rocks std value: 0.005110338105796482
metal std value: 0.006885922280666916
p-value 0.0577975369898366
we accept null hypothesis
rocks mean value: 0.006053125
metal mean value: 0.006929729729729728
rocks std value: 0.0036575536315559803
metal std value: 0.0059310238636426
p-value 0.21267788508677654
we accept null hypothesis
35 of 60 features are significantly different
```

Figure 2: Code used in the statistical test

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] NAC Cressie and HJ Whitford. How to use the two sample t-test. *Biometrical Journal*, 28(2):131–148, 1986.