Solution to Examination: Programming for Data Science (ID2214)

Henrik Boström

April 17, 2019

Part I (Theory, 10 points)

1a. Methodology, 2 points

Setting aside a larger part of the data for training a model will typically result in a more accurate model, which is less affected by the (random) choice of training instances, compared to when using a smaller training set. On the other hand, a larger test set will lead to that the variance of the performance estimate caused by the actual sample of test instances is reduced. Hence, some suitable compromise between a large training set and a large test set needs to be found.

1b. Data preparation, 2 points

Equal-sized binning will result in bins for which the observed frequencies are approximately the same. If used as a pre-processing step prior to using e.g., naïve Bayes, i.e., to transform numeric features into categoric, then equal-sized bins may be more informative than equal-width bins, as the latter may in some cases risk putting most of the observations into the same bin, hence not allowing the learningg algorithm to discriminate between the classes using the discretized feature. Equal-width binning may on the other hand be preferred to equal-sized binning, if we would like to visually inspect how feature values are distributed, e.g., if they seem to follow the normal distribution or not.

1c. Performance metrics, 2 points

If we would be interested in finding out which days are expected to be the warmest, e.g., when planning some outdoor activities, then we are more interested in the correlation between the predicted and actual temperatures, rather than being interested primarily in the absolute temperatures. The trained model with a poorer MSE may hence be more useful for this purpose than the default model, if the correlation coefficient of the former is positive (while this is not the case for the default model).

1d. Decision trees, 2 points

If numeric features are discretized prior to tree generation using (equal-sized or equal-width) binning, then the relative frequencies in the bins are not affected by

min-max normalization. Hence, the resulting tree will not be affected. Similarly, as normalization does not affect the relative order of the values, then the binary partitioning that can be obtained from the normalized and non-normalized feature, respectively, are the same, and again the resulting tree will not be affected (other than that the actual split values used will be different).

1e. Association rule mining, 2 points

A rule with a high confidence may have a very low support, and hence may not be very accurate when applied to independent test instances (not included in the database from which the rules were generated). For example, a rule with a confidence of 100% may have a support of only one instance, and hence the conclusion may only hold in 50% of the test cases, while a rule with slightly lower confidence, but much higher support, can be expected to be more correct on independent test instances. Hence, confidence alone (without a sufficiently high support) is not necessarily a meaningful evaluation criterion.

Part II (Programming, 20 points)

2a. Data preparation, 10 points

```
def select_features(df,no_features):
   class_labels = df["CLASS"]
   result = [(evaluate(df[feature],class_labels),feature)
                for feature in df.columns if feature != "CLASS"]
   result.sort(reverse=True)
   selected_features = [r[1] for r in result[:no_features]]
   return df[["CLASS"]+selected_features]
```

2b. Combining models, 10 points

```
import numpy as np

def gbm(df_orig,depth,no_trees):
   df = df_orig.copy()
   regression_values = df["REGRESSION"]
   models = [regression_values.mean()]
   predictions = np.array([models[0] for r in regression_values])
   for i in range(1,no_trees+1):
      target_values = regression_values-predictions
      df["REGRESSION"] = target_values
      reg_tree = regression_tree(df,depth)
      models.append(reg_tree)
      predictions += predict(df,reg_tree)
   return models
```