solution_to_example_examination

October 2, 2020

1 Solution to Exam: Programming for Data Science (ID2214)

I hereby declare that I have developed the solutions below without communicating with any other individual or forum, and without searching for any part of the solutions online. I also declare that I have not contributed to the solutions submitted by any other student on the exam.

Date: 2020-10-02 Name: Henrik Boström Personal no. : xxxxxx-xxxx

1.1 Part I

1.1.1 1a

The estimated accuracy of a model will vary depending on what set of instances is used to evaluate its performance. If we consider such a set of instances to be a random sample drawn from some (unknown) target distribution, the estimated accuracy of the model will sometimes be lower and sometimes higher than the true accuracy (the accuracy of the model wrt. the target distribution). If the compared models have similar true accuracies, then the observed differences in the estimated accuracies will mainly be due to the sample we have drawn, and hence the extreme values of the observed estimated accuracies will be biased; the lowest value will be overly pessimistic and the highest value overly optimistic.

Hence, if we select the highest of these values as an estimate for the true accuracy of the best model, we will systematically be overestimating the performance. However, if one of the models is much more accurate than the others (and hence would almost always be outperforming the others independently of the sample used), then this bias will be smaller. Since we typically do not beforehand know what the true accuracies are, there is hence a high risk that the estimated accuracy is indeed too high. It should be noted though that this effect to some extent is compensated by the fact that 10-fold cross-validation is employed, which provides an estimate for the model performance when having trained on 9/10 of the data, which can be expected to be lower than when using a model trained on the entire sample.

1.1.2 1b

The estimated standard deviation is used as a denominator when z-normalization is applied, and since this will be reduced due to that missing values will be replaced by the mean, it means that values that are different from the mean will end up more far away from the mean after missing values have been imputed. As the minimum and maximum values are not affected by mean-value imputation, min-max normalization will result in the same values as before.

1.1.3 1c

When looking for the class label c that maximizes $P(c|f_1 = v_1, \ldots, f_n = v_n)$ for the features $f_1, \ldots, f_n$ and values $v_1, \ldots, v_n$, using naïve Bayes, the class prior $P(c)$ is multiplied with $\prod_{i=1}^{n} P(f_i = v_i|c)$.

Hence, all features for which $P(f = 0|c_1) = P(f = 0|c_2)$ (and hence $P(f = 1|c_1) = P(f = 1|c_2)$, since the features are binary and the probabilities sum to one) will have no effect on the predicted class probabilities, and hence the accuracy will not be affected.

However, for kNN the added features will be taken into account in the distance calculations, possibly changing the neighborhood of the test instances, and hence also result in different predictions, which may have a (positive or negative) effect on the accuracy.

1.1.4 1d

Assume a binary classification task with the class labels + and -, and that we have three positive (+) instances p1, p2, p3 and two negative (-) instances n1, n2.

If M1 predicts the following probabilities for the class +:

n1: 0.9, p1: 0.8, p2: 0.7, p3: 0.6, n2: 0.1

then the accuracy is $4/5 = 0.8$ (assuming that the label with the highest probability is predicted), while the AUC is 1/2*(0/3)+1/2*(3/3) = 0.5

If M2 predicts the following probabilities for the class +:

p1: 0.6, p2: 0.4, n1: 0.3, p3: 0.2, n2: 0.1

Then the accuracy is $3/5 = 0.6$, while the AUC is 1/2*(2/3)+1/2*(3/3) = 0.833

1.1.5 1e

When generating a random forest, only a random subset of the features are evaluated when searching for the best way to split a node in a tree. If this subset constitutes less than one percent of the original features, then the cost of growing hundred trees is expected to be less than growing a single tree while evaluating all features (as the computational cost is directly proportional to the number of evaluated features). Moreover, as the trees in the random forests are generated from bootstrap replicates, the full-grown trees can be expected to be shallower, and hence faster to compute.

1.2 Part II

1.2.1 2a

[3]:
```python
def create_one_hot(orig_df,min_freq=5):
        df = orig_df.copy()
        one_hot = {}
        for col in df.columns:
            if (df.dtypes[col] == "object") and (col != "CLASS"):
                counts = df[col].value_counts()
                values = [v for v in counts.index if counts[v] >= min_freq]
                one_hot[col] = values
                new_cols_vals = [(col+"-"+str(val),val) for val in values]
                for (new_col,val) in new_cols_vals:
                        df[new_col] = (df[col] == val).astype("float")
                df.drop(col,axis="columns",inplace=True)
        return df, one_hot
```

1.2.2 2b

```
[4]: def predict(T,i):
         return make_prediction(T,0,i)

     def make_prediction(tree,node_no,instance):
         node = tree[node_no]
         if len(node) == 1:
             prediction = node[0]
         else:
             feature,threshold,left_child_node,right_child_node,left_weight = node
             value = instance[feature]
             if np.isnan(value):
                 left_prediction = make_prediction(tree,left_child_node,instance)
                 right_prediction = make_prediction(tree,right_child_node,instance)
                 prediction =␣
      ↪left_weight*left_prediction+(1-left_weight)*right_prediction
             elif value <= threshold:
                 prediction = make_prediction(tree,left_child_node,instance)
             else:
                 prediction = make_prediction(tree,right_child_node,instance)
         return prediction
```