

```
import java.awt.BorderLayout;
import java.io.IOException;
import java.util.Random;
import java.util.StringTokenizer;
import java.util.concurrent.TimeUnit;
import javax.swing.*;
import weka.attributeSelection.BestFirst;
import weka.attributeSelection.CfsSubsetEval;
import weka.attributeSelection.CorrelationAttributeEval;
import weka.attributeSelection.GainRatioAttributeEval;
import weka.attributeSelection.GreedyStepwise;
import weka.attributeSelection.InfoGainAttributeEval;
import weka.attributeSelection.PrincipalComponents;
import weka.attributeSelection.Ranker;
import weka.attributeSelection.ReliefFAttributeEval;
import weka.attributeSelection.SymmetricalUncertAttributeEval;
import weka.attributeSelection WrapperSubsetEval;
import weka.classifiers.Evaluation;
import weka.classifiers.bayes.*;
import weka.classifiers.evaluation.ThresholdCurve;
import weka.classifiers.trees.*;
import weka.classifiers.functions.*;
import weka.classifiers.functions.supportVector.PolyKernel;
import weka.classifiers.functions.supportVector.RBFKernel;
import weka.classifiers.lazy.IBk;
import weka.clusterers.ClusterEvaluation;
import weka.clusterers.EM;
import weka.clusterers.*;
import static weka.clusterers.HierarchicalClusterer.TAGS_LINK_TYPE;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.SelectedTag;
import weka.core.Utils;
import weka.core.converters.ArffSaver;
import weka.core.converters.ConverterUtils.DataSource;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.Remove;
import weka.gui.visualize.*;
```

```
public class Stock {

    public static void main(String[] args) throws Exception {
        String ticker = "";
        String[] tk = {"Shbhran", "Kpars", "Valiz"};
        String src = "-.arff";
        String dir = "D:\\DU\\Data";
        int shft = 5;
        if (args.length > 0) {
            if (args[0] != null) {
                src = args[0];
            }
        }
    }
}
```

```

    }
  }
  if (args.length > 1) {
    if (args[1] != null) {
      dir = args[1];
    }
  }
  if (args.length > 2) {
    if (args[2] != null) {
      ticker = args[2];
    }
  }
  if (args.length > 3) {
    if (args[3] != null) {
      shft = Integer.parseInt(args[3]);
    }
  }
  dir += "\\";
  int i = 0;
  int k = 0;
  ticker = tk[k];
  src = ticker + src;
  shft = 1;

  cls(dir, src, i, shft);
}

```

```

static void cls(String dir, String src, int alg, int shft) {
  try {

```

```

    Evaluation eval = null;
    double maxcorrect = 0;
    double maxerr = 1;
    double besttrsh = 1;

```

```

    Instances train1 = DataSource.read(dir + src);
    Remove rm = new Remove();

```

```

    rm.setAttributeIndices("1");
    rm.setInputFormat(train1);
    train1 = Filter.useFilter(train1, rm);

```

```

    Instances train = train1;

```

```

// *****

```

```

    if (alg < 5) {
      train = useFilter(train1, besttrsh, true, alg);
    }

```

```

    train.setClassIndex(train.numAttributes() - 1);

```

```

    int n = 1;

```

```

double sumpercent = 0, tp = 0, tn = 0, fp = 0, fn = 0, aur = 0, aup = 0;
double sumkapa = 0, sumrootErr = 0;
for (int i = 0; i < n; i++) {

    RandomForest cls = new RandomForest();
    cls.setBagSizePercent(100);
    cls.setNumIterations(100);
    cls.setBatchSize("100");
    cls.setNumDecimalPlaces(2);
    cls.setNumExecutionSlots(1);
    cls.setSeed(1);

    cls.buildClassifier(train);

    eval = new Evaluation(train);

    eval.crossValidateModel(cls, train, 10, new Random(1));

    int classIndex = 0;

// *****
    sumpercent += eval.correct() / (eval.incorrect() + eval.correct()) * 100;
    tp += eval.truePositiveRate(classIndex);
    tn += eval.trueNegativeRate(classIndex);
    fp += eval.falsePositiveRate(classIndex);
    fn += eval.falseNegativeRate(classIndex);
    aur += eval.areaUnderROC(classIndex);
    aup += eval.areaUnderPRC(classIndex);
    sumkapa += eval.kappa();
    sumrootErr += eval.rootRelativeSquaredError();
    System.out.println("Best Trsh:" + besttrsh + "\n maxcorrect: " + maxcorrect);
    System.out.println(eval.toMatrixString());
    System.out.println(eval.toSummaryString("\nResults\n=====\n", true));
    System.out.println(eval.correct() / (eval.incorrect() + eval.correct()) * 100 + "%");

}
    System.out.println("Mean Percent: " + sumpercent / n + "%");
    System.out.println("Mean Kapa: " + sumkapa / n + "%");
    System.out.println("Mean Root relative squared outor : " + sumrootErr / n + "%");

} catch (Exception e) {
    e.printStackTrace();
}
}

protected static Instances useFilter(Instances data, double trsh, boolean rs, int alg) throws
Exception {
    weka.filters.supervised.attribute.AttributeSelection filter = new
weka.filters.supervised.attribute.AttributeSelection();

    CfsSubsetEval eval1 = new CfsSubsetEval();
    BestFirst search1 = new BestFirst();

```

```

filter.setEvaluator(eval1);
filter.setSearch(search1);
filter.setInputFormat(data);
Instances newData = null, nData = null;
if (alg == 0) {
    CfsSubsetEval eval = new CfsSubsetEval();
    BestFirst search = new BestFirst();
    filter.setEvaluator(eval);
    filter.setSearch(search);
    filter.setInputFormat(data);
    newData = Filter.useFilter(data, filter);
    if ((newData.numAttributes() - 1) < 1) {
        newData = data;
    }
    if (rs) {

        for (int i = 0; i < newData.numAttributes() - 1; i++) {
            System.out.println(newData.attribute(i).name());
        }

        System.out.println(search.toString() + "\n" + eval.toString() + "\n Data Attributes: " +
(data.numAttributes() - 1) + "\n  New Data Attributes: " + (newData.numAttributes() - 1));
    }
} else if (alg == 1) {
    InfoGainAttributeEval eval = new InfoGainAttributeEval();
    Ranker search = new Ranker();
    search.setThreshold(trsh);
    filter.setEvaluator(eval);
    filter.setSearch(search);
    filter.setInputFormat(data);
    newData = Filter.useFilter(data, filter);
    if ((newData.numAttributes() - 1) < 1) {
        newData = data;
    }

    int j = 0;

    if (rs) {

        for (int i = 0; i < newData.numAttributes() - 1; i++) {
            for (j = 0; j < data.numAttributes() - 1; j++) {
                if (newData.attribute(i).name().equals(data.attribute(j).name())) {
                    break;
                }
            }
            System.out.println(newData.attribute(i).name() + ":" +
Math.floor(eval.evaluateAttribute(j) * 10000) / 10000);
        }

        System.out.println(search.toString() + "\n" + eval.toString() + "\n Data Attributes: " +
(data.numAttributes() - 1) + "\n  New Data Attributes: " + (newData.numAttributes() - 1));
    }
}

```

```

} // *****
else if (alg == 2) {
    GainRatioAttributeEval eval = new GainRatioAttributeEval();
    Ranker search = new Ranker();
    search.setThreshold(trsh);
    filter.setEvaluator(eval);
    filter.setSearch(search);
    filter.setInputFormat(data);
    newData = Filter.useFilter(data, filter);
    if ((newData.numAttributes() - 1) < 1) {
        newData = data;
    }
    if (rs) {

        for (int i = 0; i < newData.numAttributes() - 1; i++) {
            int j = 0;
            for (j = 0; j < data.numAttributes() - 1; j++) {
                if (newData.attribute(i).name().equals(data.attribute(j).name())) {
                    break;
                }
            }
        }

        System.out.println(newData.attribute(i).name());//+": "+Math.floor(eval.evaluateAttribute(j)*10000)
        /10000);
    }

    System.out.println(search.toString() + "\n" + eval.toString() + "\n Data Attributes: " +
    (data.numAttributes() - 1) + "\n New Data Attributes: " + (newData.numAttributes() - 1));
}
} // *****
else if (alg == 3) {
    CorrelationAttributeEval eval = new CorrelationAttributeEval();
    Ranker search = new Ranker();
// search.setNumToSelect(22);
    search.setThreshold(trsh);
    filter.setEvaluator(eval);
    filter.setSearch(search);
    filter.setInputFormat(data);
    newData = Filter.useFilter(data, filter);
    if ((newData.numAttributes() - 1) < 1) {
        newData = data;
    }
    if (rs) {

        for (int i = 0; i < newData.numAttributes() - 1; i++) {
            int j = 0;
            for (j = 0; j < data.numAttributes() - 1; j++) {
                if (newData.attribute(i).name().equals(data.attribute(j).name())) {
                    break;
                }
            }
        }
        System.out.println(newData.attribute(i).name() + ":" +

```

```

Math.floor(eval.evaluateAttribute(j) * 10000) / 10000);
    }

        System.out.println(search.toString() + "\n" + eval.toString() + "\n Data Attributes: " +
(data.numAttributes() - 1) + "\n  New Data Attributes: " + (newData.numAttributes() - 1));
    }
} // *****
else if (alg == 4) {
    SymmetricalUncertAttributeEval eval = new SymmetricalUncertAttributeEval();
    Ranker search = new Ranker();
    search.setThreshold(trsh);
    filter.setEvaluator(eval);
    filter.setSearch(search);
    filter.setInputFormat(data);
    newData = Filter.useFilter(data, filter);
    if ((newData.numAttributes() - 1) < 1) {
        newData = data;
    }
    if (rs) {

        for (int i = 0; i < newData.numAttributes() - 1; i++) {
            int j = 0;
            for (j = 0; j < data.numAttributes() - 1; j++) {
                if (newData.attribute(i).name().equals(data.attribute(j).name())) {
                    break;
                }
            }
            System.out.println(newData.attribute(i).name() + ":" +
Math.floor(eval.evaluateAttribute(j) * 10000) / 10000);
        }

        System.out.println(search.toString() + "\n" + eval.toString() + "\n Data Attributes: " +
(data.numAttributes() - 1) + "\n  New Data Attributes: " + (newData.numAttributes() - 1));
    }
}

return newData;
}
}

```