```java
package wekatest;

import java.io.File;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.evaluation.NominalPrediction;
import weka.classifiers.bayes.BayesNet;
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.bayes.NaiveBayesMultinomial;

import weka.classifiers.functions.Logistic;
import weka.classifiers.functions.MultilayerPerceptron;
import weka.classifiers.functions.SGD;
import weka.classifiers.functions.SMO;
import weka.classifiers.functions.VotedPerceptron;

import weka.classifiers.lazy.IBk;
import weka.classifiers.lazy.KStar;
import weka.classifiers.lazy.LWL;

import weka.classifiers.rules.DecisionTable;
import weka.classifiers.rules.JRip;
import weka.classifiers.rules.OneR;
import weka.classifiers.rules.PART;
import weka.classifiers.rules.ZeroR;

import weka.classifiers.trees.DecisionStump;
import weka.classifiers.trees.HoeffdingTree;
import weka.classifiers.trees.J48;
import weka.classifiers.trees.LMT;
import weka.classifiers.trees.RandomForest;
import weka.classifiers.trees.RandomTree;
import weka.classifiers.trees.REPTree;

import weka.core.Instances;
import weka.core.converters.CSVLoader;
import weka.filters.Filter;
import weka.filters.unsupervised.attribute.NumericToNominal;
import weka.core.FastVector;

public class WekaTest {

    public static Evaluation classify(Classifier model,
            Instances trainingSet, Instances testingSet) throws Exception {

        Evaluation evaluation = new Evaluation(trainingSet);

        model.buildClassifier(trainingSet);
        evaluation.evaluateModel(model, testingSet);

        return evaluation;
    }

    public static double calculateAccuracy(FastVector predictions) {
        double correct = 0;
        for (int i = 0; i < predictions.size(); i++) {
            NominalPrediction np = (NominalPrediction) predictions.elementAt(i);
            if (np.predicted() == np.actual()) {
                correct++;
            }
        }

        return 100 * correct / predictions.size();
    }

    public static Instances[][] crossValidationSplit(Instances data, int numberOfFolds) {
        Instances[][] split = new Instances[2][numberOfFolds];

        for (int i = 0; i < numberOfFolds; i++) {
            split[0][i] = data.trainCV(numberOfFolds, i);
            split[1][i] = data.testCV(numberOfFolds, i);
        }

        return split;
    }

    public static void main(String[] args) throws Exception {

        CSVLoader loader_train = new CSVLoader();
        loader_train.setSource(new File(
                "EECS738_Train.csv"));

        CSVLoader loader_test = new CSVLoader();
         loader_test.setSource(new File(
                 "EECS738_Test.csv"));

        // Have weka process each buffer as a "data instance"
        Instances data_train = loader_train.getDataSet();
        Instances data_test = loader_test.getDataSet();

        // Convert the class attribute to nominal
        NumericToNominal m_NumericToBinary = new NumericToNominal();
        m_NumericToBinary.setInputFormat(data_train);
        m_NumericToBinary.setAttributeIndices("2");
        data_train = Filter.useFilter(data_train, m_NumericToBinary);

        // Tell weka that the second column contains the class labels
        data_train.setClassIndex(1);
        data_test.setClassIndex(1);

        // Do 10-split cross validation
        Instances[][] split = crossValidationSplit(data_train, 10);

        // Separate split into training and testing arrays
        Instances[] trainingSplits = split[0];
        Instances[] testingSplits = split[1];

        // Use a set of classifiers
        Classifier[] models = {
            new BayesNet(),
            new NaiveBayes(),
            new NaiveBayesMultinomial (),

            new Logistic(),
            new MultilayerPerceptron (),
            new SGD(),
            new SMO(),
            new VotedPerceptron (),

            new IBk(),
            new KStar(),
            new LWL(),

            new DecisionTable (),
            new JRip(),
            new OneR(),
            new PART(),
            new ZeroR(),

            new DecisionStump (),
            new HoeffdingTree (),
            new J48(),
            new LMT(),
            new RandomForest (),
            new RandomTree (),
            new REPTree()
        };

        // Run for each model
        for (int i = 0; i < models.length; i++) {

            // Collect every group of predictions for current model
            FastVector predictions = new FastVector();

            // For each training-testing split pair, train and test the classifier
            for (int j = 0; j < trainingSplits.length; j++) {
                Evaluation validation = classify(models[i], trainingSplits[j], testingSplits[j
                ]);
                predictions.appendElements (validation.predictions ());
            }

            // Calculate overall accuracy of current classifier on all splits
            double accuracy = calculateAccuracy (predictions);

            System.out.println("Accuracy of " + models[i].getClass().getSimpleName () + ": "
                    + String.format("%.2f%%", accuracy)
                    + "\n---------------------------------" );
        }
    }
}
```