

Equipe

“Os Asdrubal”




INTERFACES

Alexandre Tamaoki – Jhonatas Santos – Marco Martins

MACHINE LEARNING PARA CLASSIFICAÇÃO

Nesta Interface, temos métodos padrões de Machine Learning, utilizados para treinar o modelo sobre uma base de dados e predizer o resultado para alguns dados inseridos.



```
import weka.core.Instances;
import weka.core.Instance;

public interface IClassificador {
    /*Metodo de Inicializacao do classificador.
    (Recomenda-se a arvore weka.classifiers.trees.J48).*/
    public void construaClassificador();
    /*Metodo para Treinar o modelo com os dados da base.*/
    public void fit();
    /*Metodo para predizer n dados entrados na forma de uma matriz.
    Retorna os targets de cada um dos dados em um vetor.*/
    public String[] predict(Instances test_data);
    /*Metodo para predizer 1 dado.
    Retorna o target do dado inserido*/
    public String predict(Instance test_data);
    /*Metodo utilizado para definir os dados de treino do classificador*/
    public void setInstances(Instances instancias);
}
```

SPLIT NA BASE DE DADOS

Aplicações:

1. Setar targets como 1 ou 0, gerando N diferentes datasets para cada diagnóstico;
2. Seleção de variáveis mais eficientes para realizar menos perguntas ao paciente.

```
import weka.core.Instances;


public interface IDataSplit{
    /*Método para definir a base de dados a ser Splitada*/
    public void setInstances(Instances instancias);
    /*Método que retorna os diferentes targets possíveis*/
    public String[] getDifTargets();
    /*Método que realiza os Splits da base de dados*/
    public void setSplit();
    /*Método que retorna a base de dados com o target
    selecionado (na posição i do vetor de DifTargets)*/
    public Instances getSplit(int i);
}
```



AVALIAR N ATRIBUTOS

Probabilidade de n classificadores serem verdadeiros dado os valores de i atributos aplicados simultaneamente nos n classificadores.

```
public interface IEvaluationNAttributes {  
    public void setInstances(String[][] instances);  
    public void setAttributes(String[] attributes);  
    public void insertClassifierEval(String classifier);  
    public void removeClassifierEval(String classifier);  
    public void insertAttributeEval(String attribute, String value);  
    public void removeAttributeEval(String attribute);  
    public String[][] eval();  
    public String[][] evalAttribute(String attribute, String value);  
    public String[][] evalWithNewAttribute(String attribute, String value);  
}
```



OTIMIZAR INSTÂNCIAS



Substituição de valores nas instâncias: o principal ganho é a capacidade de analisar um conjunto de instâncias e realizar automaticamente substituições de valores mantendo um mínimo definido de precisão na classificação posterior.

```
public interface IOptimizedDataSet extends IDataSource, ITableProducer {  
    public void optimizeAttribute(String attribute, double minAccuracy);  
    public void optimizeAllAttributes(double minAccuracy);  
  
    public void attributeReplaceValue(String attribute, String search, String replacement);  
    public void classifiersReplaceValue(String search, String replacement);  
    public void attributeReplaceLimits(String attribute, double lessThan, double moreThan, String replacement);  
    public void classifiersReplaceLimits(double lessThan, double moreThan, String replacement);  
    public void attributesReplaceValue(String search, String replacement);  
    public void attributesReplaceLimits(double lessThan, double moreThan, String replacement);  
}
```



SPLIT DO CONJUNTO DE TREINO

É usado em conjunto com o classificador para separar os dados de treino em um subconjunto de treino e teste, e assim poder achar o melhor modelo para a classificação.

```
public interface ITrain_test_split{ //Divide o conjunto principal de treino em um novo conjunto de treino e validação
    public int[][] x_train(); //Parte do conjunto de treino que foi dividido
    public int[][] x_val(); // Parte do conjunto de treino que foi dividido e separado para validação
    public String[] y_train(); // Rotulo do conjunto de treino dividido
    public String[] y_val(); // Rotulo do conjunto de treino dividido para validação
}
```



GRÁFICO DE BARRAS

Cria um gráfico de barras que pode ser utilizado para fazer histograma sobre sintomas e doenças, ou reportar os resultados do classificador.

```
import org.jfree.data.category.CategoryDataset;  
|  
public interface IGraphicsbar { //Cria gráfico de barra  
    public CategoryDataset createDataSet(ArrayList<String> Dados); //criar o dataset  
    public JFreeChart createBarChart(CategoryDataset dataSet); //criar o grafico de barras  
    public ChartPanel criarGrafico(ArrayList<String> Dados); // criar o grafico completo  
}
```

