|  |  |
| --- | --- |
| **ØµÙØ±Ø© Ø°Ø§Øª ØµÙØ©**  **تطبيق تقانات التنقيب عن البيانات لأمراض القلب**  **Naïve Bayes Classification** | **اعداد الطلاب:**  قسورة عيسى MWS\_ADM\_C3\_Qaswarah\_84545  فهد سليمان  MWT\_WDE\_C1\_Fahd\_86660  **للدكتور: باسل الخطيب**  ADM & WDE - S18 |

**أولاً: برمجة خوارزمية Bayes:**

اعتمدنا في البداية على نقل البيانات من ملف Excel الى قاعدة بيانات MS-Sql server حيث نستطيع التعامل مع قاعدة البيانات بسهولة.

**شرح سريع يوضح آلية عمل الخوارزمية المبرمجة:**

* يتم جلب بيانات الجدول وملئها في DataTable (كامل البيانات)
* يتم تدريب البيانات واستخراج المعارف منها حيث يتم على الشكل التالي
* بناء dataset
* انشاء table باسم Gaussian في DataSet وسيكون مخصص للأعمدة ذات القيم الرقمية
* انشاء table باسم Bayessian في DataSet مخصص للأعمدة النصية
* تعبئة الجدول Gaussian بمعلومات Mean & Variance عن الحقول الرقمية
* تعبئة الجدول Bayessian بالقيم الخاصة "عدد مرات التكرار في كل حالة/العدد الكلي للحالة"
* يتم حساب حالة جديدة عن طريق إجرائية على الشكل التالي
* يتم تمرير البيانات الجديدة في table
* يتم قراءة بيانات الجدول على حدى واستخراج القيم المطلوبة من Gaussian & Bayessian table
* يتم حساب القيم المقروءة وإعادة النتيجة للإظهار

**شرح الكود البرمجي:**

تم بناء الكود على ملفين:

Classifier.cs

public class Classifier

{

انشاء DataSet//

public DataSet dataSet = new DataSet();

public DataSet DataSet

{

get { return dataSet; }

set { dataSet = value; }

}

الإجرائية الخاصة بتدريب البيانات حيث يتم ادخال الجدول اليها //

public void TrainClassifier(DataTable table)

{

dataSet.Tables.Add(table);

انشاء الجدولين Gaussian & Bayessian

//table

DataTable GaussianDistribution = dataSet.Tables.Add("Gaussian");

GaussianDistribution.Columns.Add(table.Columns[0].ColumnName);

DataTable BayessianDistribution = DataSet.Tables.Add("Bayessian");

BayessianDistribution.Columns.Add(table.Columns[0].ColumnName);

قراءة أعمدة لإنشاء الاعمدة للجدولين السابقين

//columns

for (int i = 1; i < table.Columns.Count; i++)

في حال كان العمود من النمط double

{

if (table.Columns[i].DataType.Name.ToString() == "Double")

{

GaussianDistribution.Columns.Add(table.Columns[i].ColumnName + "Mean");

GaussianDistribution.Columns.Add(table.Columns[i].ColumnName + "Variance");

سوف ينشئ في الجدول Gaussian عمودين Mean & Variance لكل عمود يتم قراءته بنمط Double

}

else if (table.Columns[i].DataType.Name.ToString() == "String")

سيتم الاستعلام عن القيم التي يحتويها هذا العمود ويعيد قيم كل قيمة متوفرة في العمود مع عدد التكرار

{

var Slst = (from myRow in table.AsEnumerable()

orderby myRow.Field<string>(table.Columns[i]) ascending

group myRow by myRow.Field<string>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

foreach (var itm in Slst)

{

انشاء عمود باسم العمود الأساسي + اسم القيمة المتكررة. وذلك لكل قيمة متكررة

if ((string)itm.Name != "")

{

BayessianDistribution.Columns.Add(table.Columns[i].ColumnName + " " + itm.Name);

}

في حال كان نمط البيانات من النوع boolean سوف ينشئ أعمدة بعدد القيم التي سيتم حسابها ضمن الكود

}

}

else if (table.Columns[i].DataType.Name.ToString() == "Boolean")

{

var Blst = (from myRow in table.AsEnumerable()

orderby myRow.Field<bool>(table.Columns[i]) ascending

group myRow by myRow.Field<bool>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

انشاء عمود باسم العمود الأساسي + اسم القيمة المتكررة. وذلك لكل قيمة متكررة

foreach (var itm in Blst)

{

BayessianDistribution.Columns.Add(table.Columns[i].ColumnName + " " + itm.Name);

}

}

بعد أن تم انشاء أعمدة الجدولين Gaussian & Bayessian سيتم حساب قيم الخلايا

}

في هذا الاستعلام سيعيد قيم العمود الأول وهوDisease وكل قيمة عدد مرات تكرارها

//calc data

var results = (from myRow in table.AsEnumerable()

group myRow by myRow.Field<string>(table.Columns[0].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

for (int j = 0; j < results.Count; j++)

من خلال حلقة For سوف يتم ادخال قيم صفوف الجدولين لكل حالة من حالات Disease

{

DataRow row = GaussianDistribution.Rows.Add();

row[0] = results[j].Name;

int a = 1;

for (int i = 1; i < table.Columns.Count; i++)

سيتم حساب mean وVariance وادخالهم للجدول

{

if (table.Columns[i].DataType.Name.ToString() == "Double")

{

row[a] = Helper.Mean(SelectRows(table, i, string.Format("{0} = '{1}'", table.Columns[0].ColumnName, results[j].Name)));

row[++a] = Helper.Variance(SelectRows(table, i, string.Format("{0} = '{1}'", table.Columns[0].ColumnName, results[j].Name)));

a++;

}

}

}

var Dresults = (from myRow in table.AsEnumerable()

group myRow by myRow.Field<string>(table.Columns[0].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

for (int j = 0; j < Dresults.Count; j++)

{

DataRow row = BayessianDistribution.Rows.Add();

double r = Convert.ToDouble(results[j].Count) / (Convert.ToDouble(results[0].Count) + Convert.ToDouble(results[1].Count));

بالنسبة لجدول Bayessian يتم استعلام عدد مرات حالات Disease ومن ثم خلال حلقة for يتم استعلام الاعمدة النصية واستعلام القيم المتكررة وعدد مرات تكرارها ويتم ادخال لكل عامود عدد مرات التكرار / العدد الكلي لحالة disease

row[0] = r;

int b = 1;

for (int i = 1; i < table.Columns.Count; i++)

{

if (table.Columns[i].DataType.Name.ToString() == "String")

{

var Cresults = (from myRow in table.AsEnumerable()

orderby myRow.Field<string>(table.Columns[i]) ascending

group myRow by myRow.Field<string>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

var PNresults = (from myRow in table.AsEnumerable()

orderby myRow.Field<string>(table.Columns[i]) ascending

where myRow.Field<string>(table.Columns[0].ColumnName.ToString()) == Dresults[j].Name

group myRow by myRow.Field<string>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

foreach (var item in PNresults)

{

if ((string)item.Name != "")

{

float R = (float)item.Count / Dresults[j].Count;

row[b] = R.ToString();

b++;

}

}

}

بالنسبة لأعمدة Boolean سوف يتم بنفس طريقة String

else if (table.Columns[i].DataType.Name.ToString() == "Boolean")

{

var Cresults = (from myRow in table.AsEnumerable()

orderby myRow.Field<bool>(table.Columns[i]) ascending

group myRow by myRow.Field<bool>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

var Bresults = (from myRow in table.AsEnumerable()

orderby myRow.Field<bool>(table.Columns[i]) ascending

where myRow.Field<string>(table.Columns[0].ColumnName.ToString()) == Dresults[j].Name

group myRow by myRow.Field<bool>(table.Columns[i].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

foreach (var item in Bresults)

{

double R = (double)item.Count / Dresults[j].Count;

row[b] = R.ToString();

b++;

}

}

}

}

}

إجرائية حساب الحالة المدخلة ويتم ادخال اليها جدول من عمودين

عمود يحوي أسماء الاعمدة والثاني يحوي القيم المدخلة

public string Classify(DataTable table2)

{

Dictionary<string, double> score = new Dictionary<string, double>();

var results = (from myRow in dataSet.Tables[0].AsEnumerable()

group myRow by myRow.Field<string>(dataSet.Tables[0].Columns[0].ColumnName) into g

select new { Name = g.Key, Count = g.Count() }).ToList();

var Cresults = (from myRow in table2.AsEnumerable()

في هذين الاستعلامين سيتم تجهيز البيانات المدخلة للإجرائية الأول يعيد Disease كل حالة لها.

الثاني ينظم الجدول في list يحوي كل سطر عنوان العمود والقيمة

حلقة for سيتم تمريرها على كامل البيانات المدخلة

select new

{

Name = myRow.Field<string>(table2.Columns[0]),

key = myRow.Field<string>(table2.Columns[1])

}).ToList();

for (int i = 0; i < results.Count; i++)

{

List<double> subScoreList = new List<double>();

subScoreList.Add(Convert.ToDouble(dataSet.Tables["Bayessian"].Rows[i][0]));

for (int f = 0; f < Cresults.Count; f++)

في البداية سوف يتم استخراج قيم Gaussian للأعمدة الرقمية وادخالها في SubScoreList

{

for (int k = 1; k < dataSet.Tables["Gaussian"].Columns.Count; k = k + 2)

{

if (dataSet.Tables["Gaussian"].Columns[k].ColumnName.ToString() == Cresults[f].Name.ToString() + "Mean")

{

double mean = Convert.ToDouble(dataSet.Tables["Gaussian"].Rows[i][k]);

double variance = Convert.ToDouble(dataSet.Tables["Gaussian"].Rows[i][k + 1]);

double result = Helper.NormalDist(Convert.ToDouble(Cresults[f].key), mean, Helper.SquareRoot(variance));

subScoreList.Add(result);

}

}

يتم استخراج قيم الجدول Bayessian للاعمدة النصية وادخالها في subScoreList

}

for (int f = 0; f < Cresults.Count; f++)

{

for (int j = 1; j < dataSet.Tables["Bayessian"].Columns.Count; j++)

{

if (dataSet.Tables["Bayessian"].Columns[j].ToString() == Cresults[f].Name.ToString() + " " + Cresults[f].key.ToString())

{

double result = Convert.ToDouble(dataSet.Tables["Bayessian"].Rows[i][j]);

subScoreList.Add(result);

}

}

}

double finalScore = 0;

foreach (double itm in subScoreList)

{

يتم تمرير قيم subScoreList في حلقة حساب ليتم حساب Probability لكل حالة مرض Disease

تخزين النتيجة في dictionary والحالة Positive او Negative

من ثم اجراء استعلام عن القيمة الكبيرة وإظهار الاسم العائد للقيمة

if (finalScore == 0)

{

finalScore = itm;

}

else

{

finalScore = finalScore \* itm;

}

}

score.Add(results[i].Name, finalScore);

}

double maxOne = score.Max(c => c.Value);

var name = (from c in score

where c.Value == maxOne

select c.Key).First();

return name;

}

#region Helper Function

public IEnumerable<double> SelectRows(DataTable table, int column, string filter)

{

List<double> \_doubleList = new List<double>();

DataRow[] rows = table.Select(filter);

for (int i = 0; i < rows.Length; i++)

{

\_doubleList.Add((double)rows[i][column]);

}

return \_doubleList;

}

public void Clear()

{

dataSet = new DataSet();

}

#endregion

}

Helper.cs يتم استخدامه بشكل ضمني في تعليمات الملف Classifier.cs

public static class Helper

{

public static double Variance(this IEnumerable<double> source)

اجراء حساب التغيير variance

{

double avg = source.Average();

double d = source.Aggregate(0.0, (total, next) => total += Math.Pow(next - avg, 2));

return d / (source.Count() - 1);

}

public static double Mean(this IEnumerable<double> source)

{

if (source.Count() < 1)

اجراء حساب المتوسط

return 0.0;

double length = source.Count();

double sum = source.Sum();

return sum / length;

}

public static double NormalDist(double x, double mean, double standard\_dev)

{

اجراء حساب التوزيع الطبيعي

double fact = standard\_dev \* Math.Sqrt(2.0 \* Math.PI);

double expo = (x - mean) \* (x - mean) / (2.0 \* standard\_dev \* standard\_dev);

return Math.Exp(-expo) / fact;

}

public static double SquareRoot(double source)

اجراء حساب الجذر

{

return Math.Sqrt(source);

}

}

**استخدام الإجرائية في الصفحة:**

* يتم تدريب الجدول باستخدام التعليمة:

Classifier classifier = new Classifier();

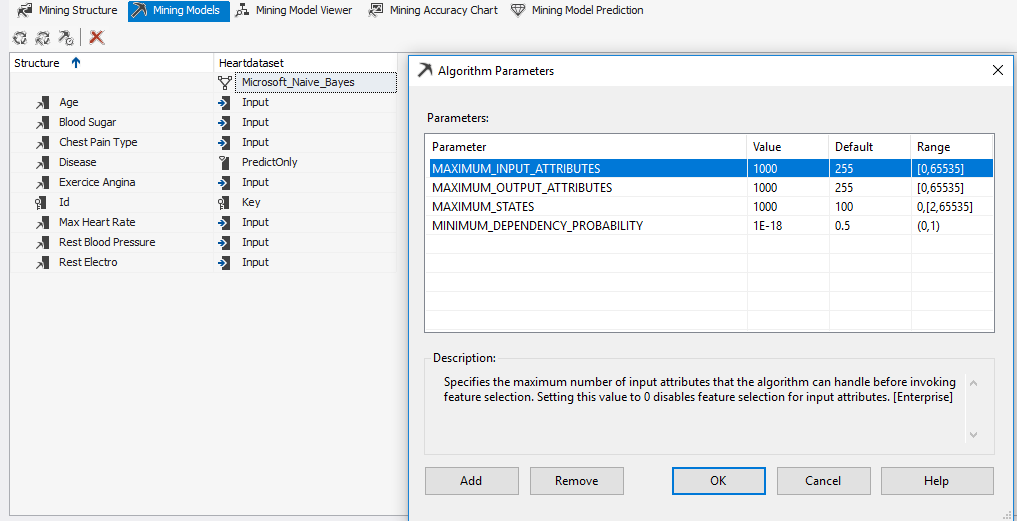
classifier.TrainClassifier(table);

* يتم استعلام نتيجة حالة جديدة باستخدام التعليمة:

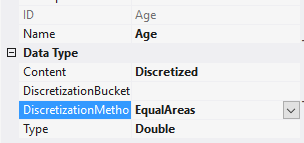
Label2.Text = classifier.Classify(table2);

**ثانياً: استخدام أدوات SQL Server:**

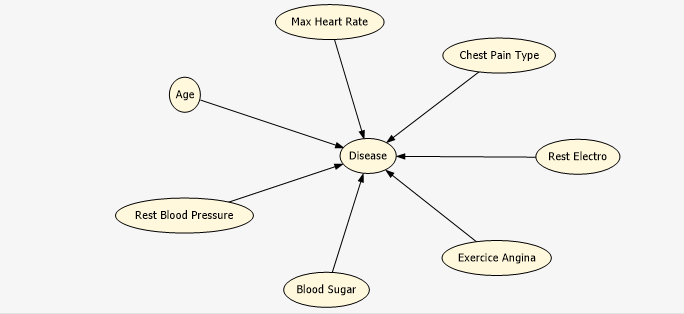
من المهم اعداد بارامترات الخوارزمية بشكل جيد كما في الصورة.



بالنسبة للأعمدة النصية فيجب تفعيل خاصية DiscretizationMethod واعطائها قيمة EqalEreas حيث ستعمل على تجزئة القيم الى مجالات عددية



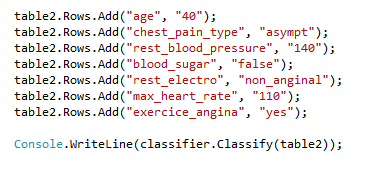
توضح الصورة التالية عرض نمط التنقيب بعد ضبط البارامترات والخاصيات للأعمدة العددية

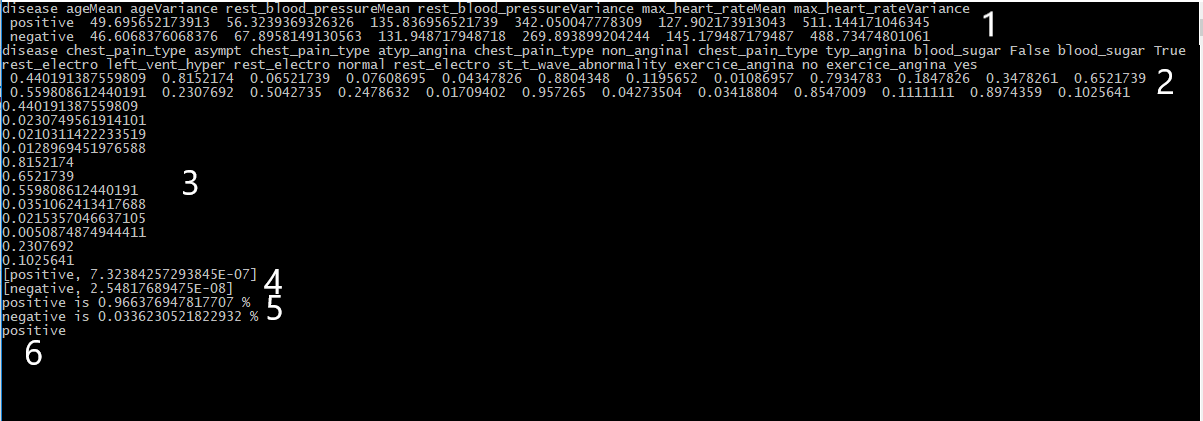


ثالثاً: المقارنة

في المقارنة اعتمدنا أن نستخدم مشروع C# للخوارزمية المبرمجة للتنفيذ وعرض قيم الجداول التي تم حسابها لتتم مقارنتها بنتائج الأداة الجاهزة

فيما يلي تنفيذ الخوارزمية على القيم التالية:



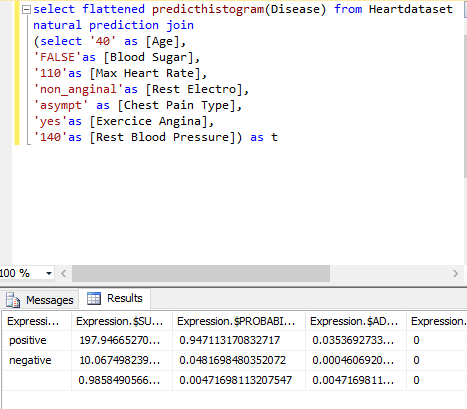


1. في التنفيذ يظهر أولاً جدول Gaussian للأعمدة العددية وكل عمود له عمودين mean & Variance وقيم الحالتين في كل سطر Disease

Positive & negative

1. الجدول الثاني Bayessian يحوي على أعمدة الجدول النصية حيث لكل عمود له عدد من الاعمدة بحسب القيم المتكررة
2. ثم يظهر ليدنا probability لكل قيمة مدخلة في الحالة الجديدة التي يتم حسابها في الحالتين positive & negative على التسلسل
3. ثم يظهر لدينا حساب كل حالة على حدى
4. يظهر لدينا normalization لكل حالة بالنسبة المئوية
5. يظهر لدينا القيمة المعادة من الإجرائية

مقارنة النتيجة السابقة مع الأداة الجاهزة:



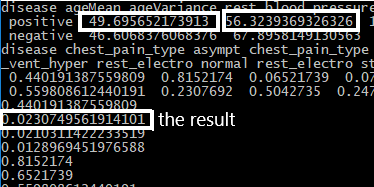
* **بالمقارنة يتوضح لدينا الجدول التالي لاحتمالية القيم Probability**

|  |  |  |  |
| --- | --- | --- | --- |
| Disease | الخوارزمية المبرمجة | الأداة الجاهزة | الفارق |
| Positive | 0.96637694 | 0.94711317 | 0.019 |
| negative | 0.03362052 | 0.04816984 | 0.014 |

* **التأكد من أن الخوارزمية تقوم بالحساب بشكل صحيح:**

سنستخدم المثال السابق للتأكد وسنقوم بحساب العمر Age والذي أدخلنا قيمته في المثال 40

في الصورة التالية نبيّن القيم المحسوبة خلال الخوارزمية للحالة positive



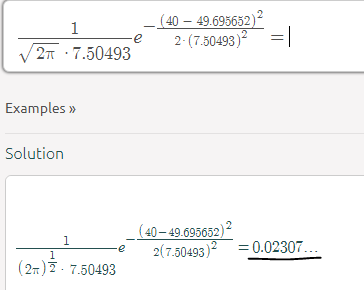
نجد أن Age Mean = 49.695652

Age Variance = 56.323936

نقوم بحساب Standard Division



نقوم بحساب Normal Distribution



يتبين أن الحساب صحيح100% حسابياً

**رابعاً: استخدام الخوارزمية المبرمجة مع جداول مختلفة:**

يمكن استخدامها مع أية جداول تحوي على أعمدة من الأنماط التالية: String & Double & Boolean وهي قابلة للتطوير لتشمل بيانات أخرى

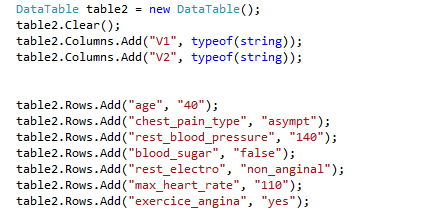
الأمور التي يجب مراعاتها لتحقيق ذلك:

* عند استعلام بيانات الجدول يجب الاستعلام بهذا الشكل:

Select [Prediction Column], [the other columns next that]…etc. from your theTable;

أي يجب استعلام حقل Predict أولاً ومن ثم بقية الاعمدة

* يجب استعمال DataTable كمساحة تخزين مؤقتة في تمرين الجدول واستنتاج القيم
* عند الاستعلام عن حالة جديدة والتنبؤ بها يجب تمريرها باستخدام datatable ويكون بهذا الشكل



حيث العمود الأول يحوي أسماء الأعمدة بشكل صحيح والعمود الثاني يحوي القيم المختبرة

* لا يهم عند الاستعلام عن حالة جديدة ترتيب الاعمدة بنفس ترتيب الجدول أو غيره.

**خامسا: التقنيات المستخدمة:**

* استخدمنا SQL Server 2014 لقاعدة البيانات Data base Engine بالإضافة بالنسبة SQL Analysis Service
* بناء الصفحة تم باستخدام Microsoft Visual Studio 2012

**المراجع المستخدمة**

* [Nave Bayes Classifier, by Milan Solanki](https://www.codeproject.com/Articles/318126/Naive-Bayes-Classifier)
* [Microsoft Documentation, Microsoft Naïve Bayes](https://docs.microsoft.com/en-us/sql/analysis-services/data-mining/microsoft-naive-bayes-algorithm?view=sql-server-2017)