**Classification**

**Classification** dalam data mining merupakan proses untuk mengidentifikasi kategori objek berdasarkan **training dataset** yang mengandung objek yang kategorinya telah diketahui. Sebagai contoh jika kita ingin mengklasifikasikan hewan berikut:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama Hewan | Alat Pernafasan | Berkembang Biak | Menyusui | Kelas (Label) |
| Belut | Insang | Bertelur | Tidak | ? |
| Gajah | Paru-Paru | Melahirkan | Ya | ? |

Dengan data training sebagai berikut:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama Hewan | Alat Pernafasan | Berkembang Biak | Menyusui | Kelas (Label) |
| Hiu | Insang | Bertelur | Tidak | Ikan |
| Pari | Insang | Bertelur | Tidak | Ikan |
| Kera | Paru-Paru | Melahirkan | Ya | Mamalia |
| Harimau | Paru-Paru | Melahirkan | Ya | Mamalia |
| Tuna | Insang | Bertelur | Tidak | Ikan |

Maka kita dapat mengkategorikan bahwa belut tergolong kelas ikan dan gajah merupakan kelas mamalia berdasarkan kemiripan ciri-ciri yang ada antara hewan yang ingin di klasifikasi dengan hewan pada data training.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama Hewan | Alat Pernafasan | Berkembang Biak | Menyusui | Kelas (Label) |
| Belut | Insang | Bertelur | Tidak | Ikan |
| Gajah | Paru-Paru | Melahirkan | Ya | Mamalua |

**Algoritma Classification**

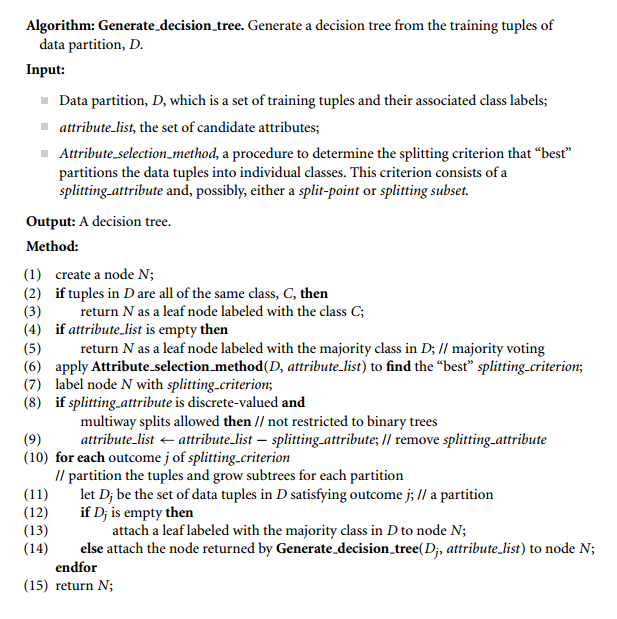
Ada beberapa algoritma yang dapat digunakan untuk melakukan *classification* yaitu:

1. **Decision Tree**

*Decision Tree* adalah sebuah struktur pohon, dimana setiap *node* pohon merepresentasikan atribut yang telah diuji, setiap cabang merupakan suatu pembagian hasil uji, dan node daun (leaf) merepresentasikan kelompok kelas tertentu. Salah satu pendekatannya untuk menghasilkan *decision tree* adalah **ID3 algorithm**. **ID3 algorithm** menggunakan *information gain* sebagai *attribute selection**measure* atau *splitting criterion*. *Information gain* dapat dihitung berdasarkan *information needed* sebelum memisahkan pada atribut dan *information needed* setelah membelah pada atribut. *Information needed* (*entropy*) adalah informasi yang diperlukan untuk mengklasifikasikan tupel dalam dataset D.

*Decision tree* akan terus membagi dataset berdasarkan atribut dengan *information gain* tertinggi hingga setiap tupel dalam dataset memiliki kelas yang sama (konvergen).

* **Algoritma**

****

**Contoh**

Manajer AllElectronic ingin membuat classifier menggunakan *decision tree algorithm* untuk mengklasifikasikan karakteristik pelanggan yang ingin membeli komputer.

Training Set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | High | No | Fair | No |
| <= 30 | High | No | Excellent | No |
| 31..40 | High | No | Fair | Yes |
| > 40 | Medium | No | Fair | Yes |
| > 40 | Low | Yes | Fair | Yes |
| > 40 | Low | Yes | Excellent | No |
| 31..40 | Low | Yes | Excellent | Yes |
| <= 30 | Medium | No | Fair | No |
| <= 30 | Low | Yes | Fair | Yes |
| > 40 | Medium | Yes | Fair | Yes |
| <= 30 | Medium | Yes | Excellent | Yes |
| 31..40 | Medium | No | Excellent | Yes |
| 31..40 | High | Yes | Fair | Yes |
| > 40 | Medium | No | Excellent | No |

* Langkah 1

Lakukan *attribute selection method* untuk menemukan *splitting criterion* terbaik. *Attribute selection method* adalah metode untuk memilih atribut dengan *information gain* tertinggi. *Information needed* (*entropy*) dapat dihitung dengan rumus berikut:

**Info (D)**

Informasi yang diperlukan untuk mengklasifikasikan tupel dalam dataset D

**M**

Jumlah kelas dalam dataset D

**Pi**

Probabilitas non-nol untuk setiap kelas

*Information needed* setelah pemisahan pada atribut A dapat dihitung dengan rumus berikut:

**InfoA (D)**

Informasi yang dibutuhkan setelah pemisahan pada atribut A untuk mengklasifikasikan tuple dalam dataset D

**V**

Jumlah nilai yang berbeda dari atribut A dalam dataset D

**|Dj|**

Jumlah item untuk setiap nilai yang berbeda dari atribut A

**|D|**

Jumlah item dalam kumpulan data

*Information gain* dapat dihitung dengan rumus berikut:

**Gain (A)**

Informasi yang diperoleh dengan memisahkan pada atribut A

**Info (D)**

Informasi yang diperlukan untuk mengklasifikasikan tupel dalam dataset D

**InfoA (D)**

Informasi yang dibutuhkan setelah pemisahan pada atribut A untuk mengklasifikasikan tuple dalam dataset D

**Information Gain** dari atribut **Age**

**Information Gain** dari atribut **Income**

**Information Gain** dari atribut **Student**

**Information Gain** dari atribut **Credit\_Rating**

* Langkah 2

Hasil dari *attribute selection method* adalah **Age** karena **Age** memiliki *information gain* tertinggi. Jadi, kita bagi kumpulan data berdasarkan **Age**.

**Age <= 30**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | High | No | Fair | No |
| <= 30 | High | No | Excellent | No |
| <= 30 | Medium | No | Fair | No |
| <= 30 | Low | Yes | Fair | Yes |
| <= 30 | Medium | Yes | Excellent | Yes |

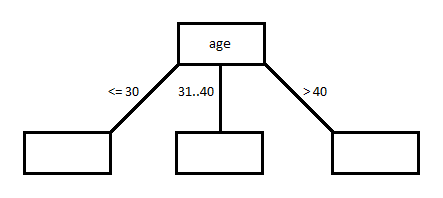
**Age 31..40**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| 31..40 | High | No | Fair | Yes |
| 31..40 | Low | Yes | Excellent | Yes |
| 31..40 | Medium | No | Excellent | Yes |
| 31..40 | High | Yes | Fair | Yes |

**Age > 40**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| > 40 | Medium | No | Fair | Yes |
| > 40 | Low | Yes | Fair | Yes |
| > 40 | Low | Yes | Excellent | No |
| > 40 | Medium | Yes | Fair | Yes |
| > 40 | Medium | No | Excellent | No |

**Tree**

****

* Langkah 3

Lakukan *attribute selection method* lagi untuk setiap cabang.

**Age <= 30**

**Information Gain** dari atribut **Income**

**Information Gain** dari atribut **Student**

**Information Gain** dari atribut **Credit\_Rating**

**Age 31…40**

* Di cabang umur 31…40, setiap tupel memiliki kelas yang sama, yaitu **Yes**. Jadi kita dapat mengira bahwa pelanggan dengan usia 31…40 berada di kelas **Yes**.

**Age > 40**

**Information Gain** dari atribut **Age**

**Information Gain** dari atribut **Income**

**Information Gain** dari atribut **Student**

**Information Gain** dari atribut **Credit\_Rating**

* Langkah 4

Di cabang **Age** <= 30, atribut dengan ***information gain*** tertinggi adalah **Student**. Jadi, kita bagi dataset berdasarkan **Student**.

**Student**: **No**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | High | No | Fair | No |
| <= 30 | High | No | Excellent | No |
| <= 30 | Medium | No | Fair | No |

**Student**: **Yes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | Low | Yes | Fair | Yes |
| <= 30 | Medium | Yes | Excellent | Yes |

Di cabang **Age** > 40, atribut dengan **information gain** tertinggi adalah **Credit\_Rating**. Jadi, kita bagi dataset berdasarkan **Credit\_Rating**.

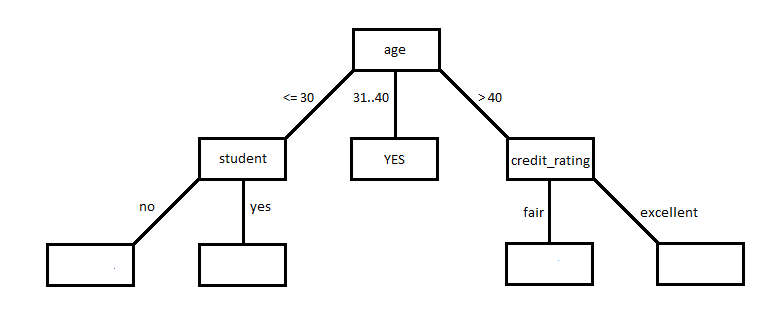
**Credit\_Rating**: **Fair**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| > 40 | Medium | No | Fair | Yes |
| > 40 | Low | Yes | Fair | Yes |
| > 40 | Medium | Yes | Fair | Yes |

**Credit\_Rating**: **Excellent**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| > 40 | Low | Yes | Excellent | No |
| > 40 | Medium | No | Excellent | No |

**Tree**



* Langkah 5

Terapkan *attribute selection method* lagi untuk setiap cabang

**Age** <= **30** dan **Student**: **No**

* Di cabang **Age** <= 30 dan **Student**: No, setiap tupel memiliki kelas yang sama, yaitu **No**. Jadi kita dapat menganggap bahwa pelanggan dengan age di bawah atau sama dengan 30 dan bukan seorang student di kelas **No**.

**Age** <= **30** dan **Student**: **Yes**

* Di cabang **Age** <= 30 dan **Student**: Yes, setiap tupel memiliki kelas yang sama, yaitu **Yes**. Jadi kita dapat menganggap bahwa pelanggan dengan age di bawah atau sama dengan 30 dan bukan seorang student di kelas **Yes**.

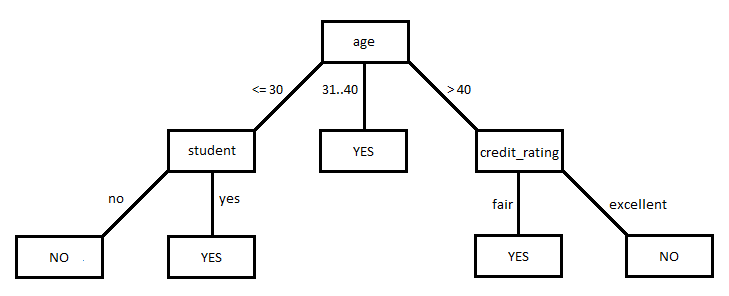
**Age** > **40** dan **Credit\_Rating**: **Fair**

* Di cabang **Age** > 40 dan **Credit\_Rating**: Fair, setiap tupel memiliki kelas yang sama, yaitu **Yes**. Jadi kita dapat mengira bahwa pelanggan dengan age lebih dari 40 dan credit rating fair berada di kelas **Yes**.

**Age** > **40** dan **credit rating**: **Excellent**

* Di cabang **Age** > 40 dan **Credit\_Rating**: Excellent, setiap tupel memiliki kelas yang sama, yaitu **No**. Jadi kita dapat menganggap bahwa pelanggan dengan age lebih dari 40 dan credit rating excellent berada di kelas **No**.

**Tree**



1. **K-NN**

K-NN (*K-Nearest Neighbour*) adalah salah satu metode *lazy learning*. Untuk mengklasifikasikan tuple X yang tidak diketahui, K-NN akan menggunakan label k-tetangga terdekat (*K-Nearest Neighbour*) untuk menentukan label tuple.

* **Algorithm**

Membutuhkan tiga hal:

* Kumpulan data training
* *Distance Metric* untuk menghitung jarak antar data yang akan diklasifikasikan dengan setiap data dalam set training
* Nilai k, jumlah tetangga terdekat

Untuk mengklasifikasikan data yang tidak berlabel:

* Hitung jarak antara data yang ingin diklasifikasikan dengan setiap data dalam set pelatihan
* Identifikasi k tetangga terdekat dengan mengambil k-objek yang memiliki jarak terdekat dari data yang akan diklasifikasikan
* Gunakan label tetangga k-terdekat untuk menentukan label data yang akan diklasifikasikan dengan mengambil label yang paling banyak muncul

**Example**

Training set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | High | No | Fair | No |
| <= 30 | High | No | Excellent | No |
| 31..40 | High | No | Fair | Yes |
| > 40 | Medium | No | Fair | Yes |
| > 40 | Low | Yes | Fair | Yes |
| > 40 | Low | Yes | Excellent | No |
| 31..40 | Low | Yes | Excellent | Yes |
| <= 30 | Medium | No | Fair | No |
| <= 30 | Low | Yes | Fair | Yes |
| > 40 | Medium | Yes | Fair | Yes |
| <= 30 | Medium | Yes | Excellent | Yes |
| 31..40 | Medium | No | Excellent | Yes |
| 31..40 | High | Yes | Fair | Yes |
| > 40 | Medium | No | Excellent | No |

Data untuk diklasifikasikan:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| <= 30 | Medium | Yes | Fair | ? |

* Langkah 1

Tentukan parameter K = jumlah tetangga terdekat. Kita asumsikan K = 3.

* Langkah 2

Hitung jarak antara data yang akan diklasifikasikan dan setiap data dalam set training. Untuk menghitung jarak, akan lebih mudah jika datanya numerik. Dalam kasus ini, data tidak numerik, jadi kita harus mengubah data tidak numerik menjadi data numerik:

* Grade untuk age:
* <= 30 : 0
* 31..40 : 1
* > 40 : 2
* Grade untuk income:
* Low : 0
* Medium : 1
* High : 2
* Grade untuk Student:
* No : 0
* Yes : 1
* Grade untuk credit\_rating:
* Fair : 0
* Excellent : 1

Hasil training set:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| 0 | 2 | 0 | 0 | No |
| 0 | 2 | 0 | 1 | No |
| 1 | 2 | 0 | 0 | Yes |
| 2 | 1 | 0 | 0 | Yes |
| 2 | 0 | 1 | 0 | Yes |
| 2 | 0 | 1 | 1 | No |
| 1 | 0 | 1 | 1 | Yes |
| 0 | 1 | 0 | 0 | No |
| 0 | 0 | 1 | 0 | Yes |
| 2 | 1 | 1 | 0 | Yes |
| 0 | 1 | 1 | 1 | Yes |
| 1 | 1 | 0 | 1 | Yes |
| 1 | 1 | 1 | 0 | Yes |
| 2 | 1 | 0 | 1 | No |

Data untuk diklasifikasikan:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **Income** | **Student** | **Credit\_rating** | **Buys\_computer (Label)** |
| 0 | 1 | 1 | 0 | ? |

* Langkah 3

Hitung jarak antara data yang akan diklasifikasikan dan setiap data dalam set pelatihan. Gunakan *Euclidean Distance* untuk menghitung jarak:

**d (p, q)**

Jarak antara objek p dan objek q

**p**

Data harus diklasifikasikan

**q**

Setiap data dalam set pelatihan

**xi**

Atribut-I dari objek p

**yi**

Atribut-I dari objek q

**t**

Jumlah atribut

* Langkah 4

Urutkan jarak dalam urutan *ascending* untuk menemukan tetangga terdekat

|  |  |
| --- | --- |
| **Distance** | **Buys\_computer (Label)** |
|  | No |
|  | Yes |
|  | Yes |
|  | Yes |
|  | No |
|  | Yes |
|  | No |
|  | Yes |
|  | Yes |
|  | Yes |
|  | Yes |
|  | Yes |
|  | No |
|  | No |

* Langkah 5

3 tetangga terdekat pertama

|  |  |
| --- | --- |
| **Distance** | **Buys\_computer (Label)** |
|  | No |
|  | Yes |
|  | Yes |

|  |  |
| --- | --- |
| **Buys\_computer (Label)** | **Percentage** |
| Yes | 2 / 3 = 0.667 |
| No | 1 / 3 = 0.333 |

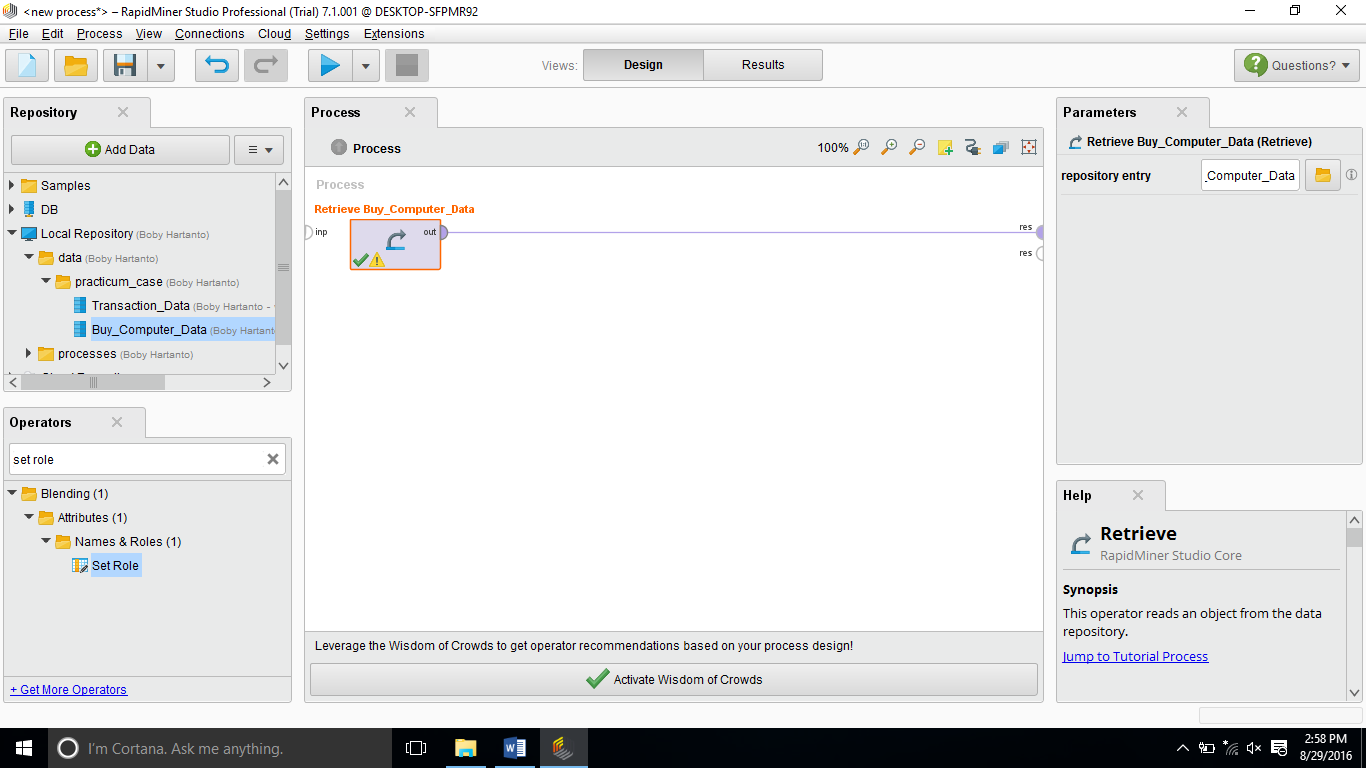
Dari table di atas kita dapat melihat bahwa dari ***k-nearest neighbours*** dari data yang ingin diklasifikasikan dengan **k** = **3**, jumlah **nearest neighbours** dengan kelas **buys\_computer** = **“yes”** lebih dari jumlah **nearest neighbours** dengan kelas **buys\_computer** = **“no”**. Jadi, kita dapat mengklasifikasikan bahwa data yang akan diklasifikasikan termasuk dalam kelas **buys\_computer** = **“yes”**.

**Practicum in Rapid Miner**

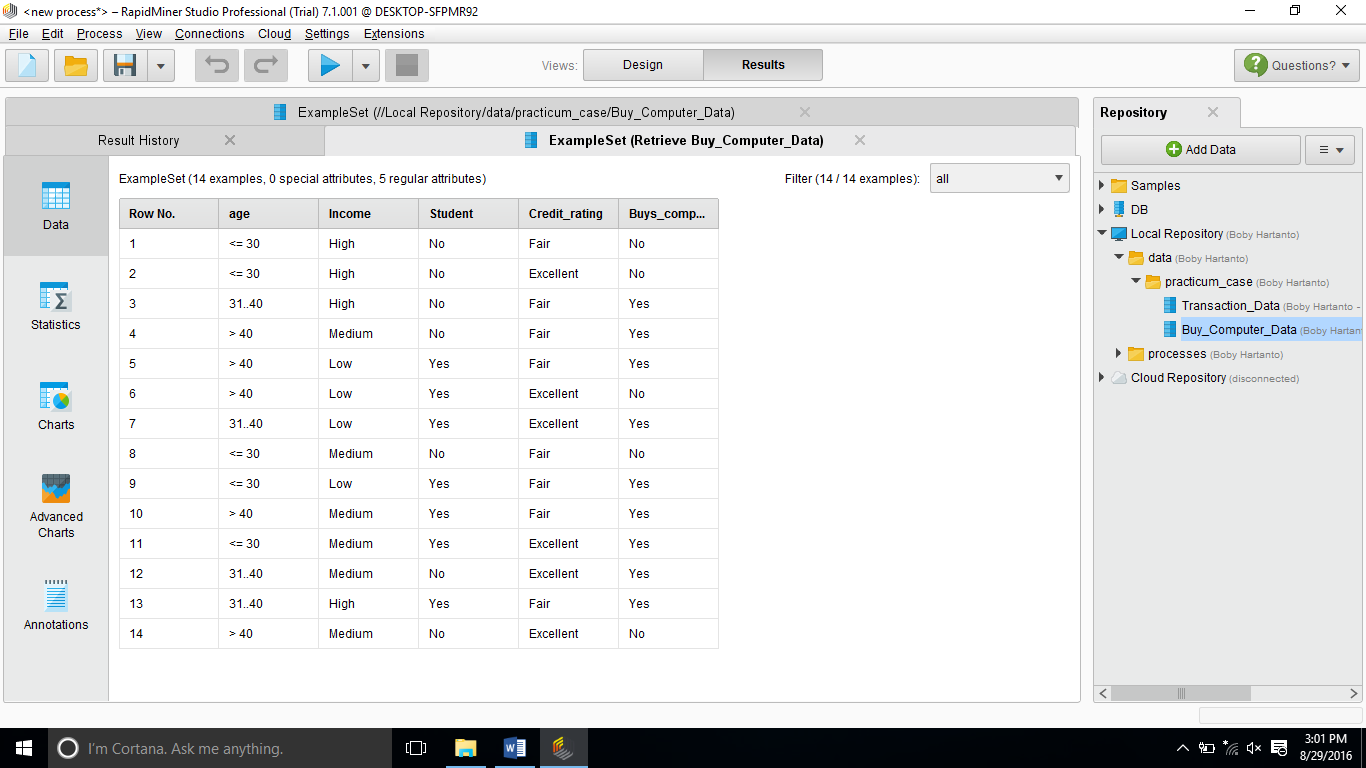
1. **Decision Tree**

* Langkah 1

Impor dan tarik Buy\_Computer\_Data ke dalam process window

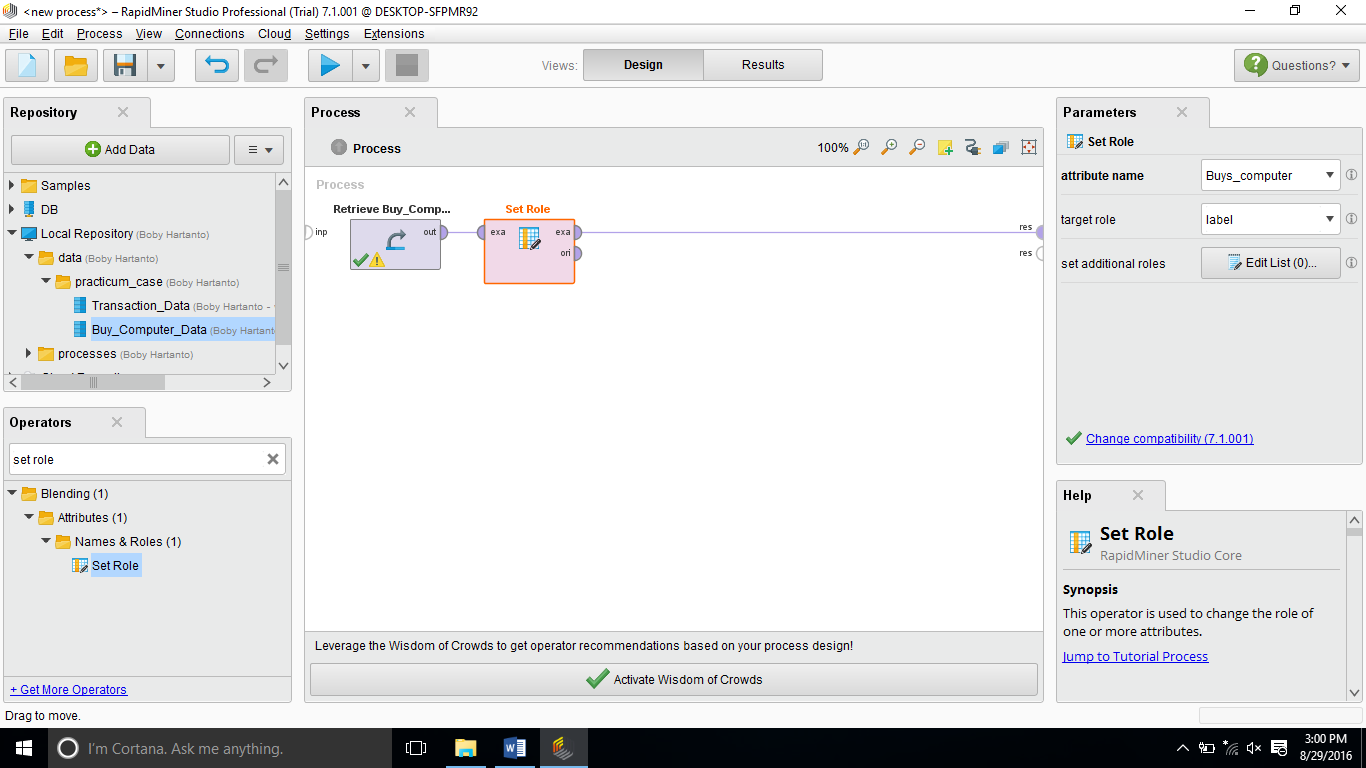


Hasil:



* Langkah 2

Gunakan operator set role untuk mengubah role kolom Buys\_Computer menjadi label



Catatan:

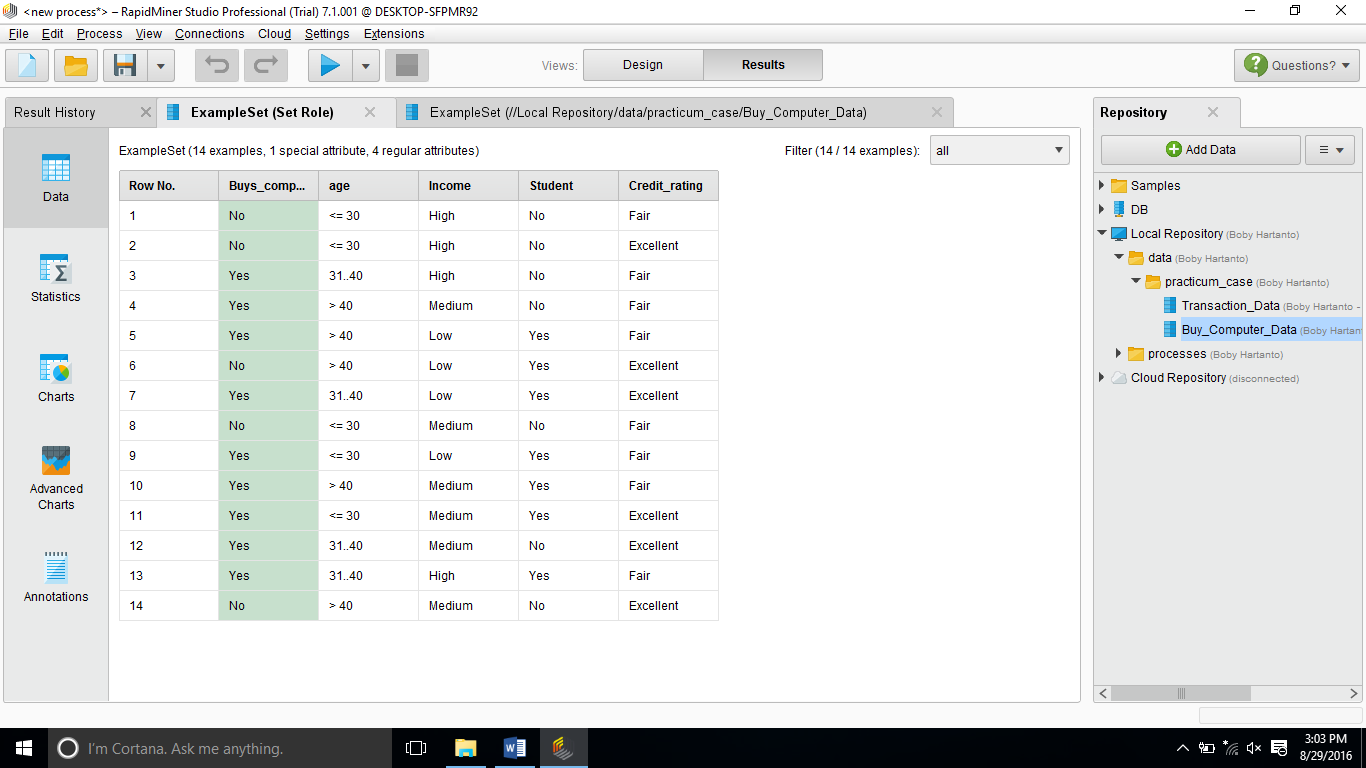
**Attribute Name**

Nama atribut yang perannya harus diubah

**Target role**

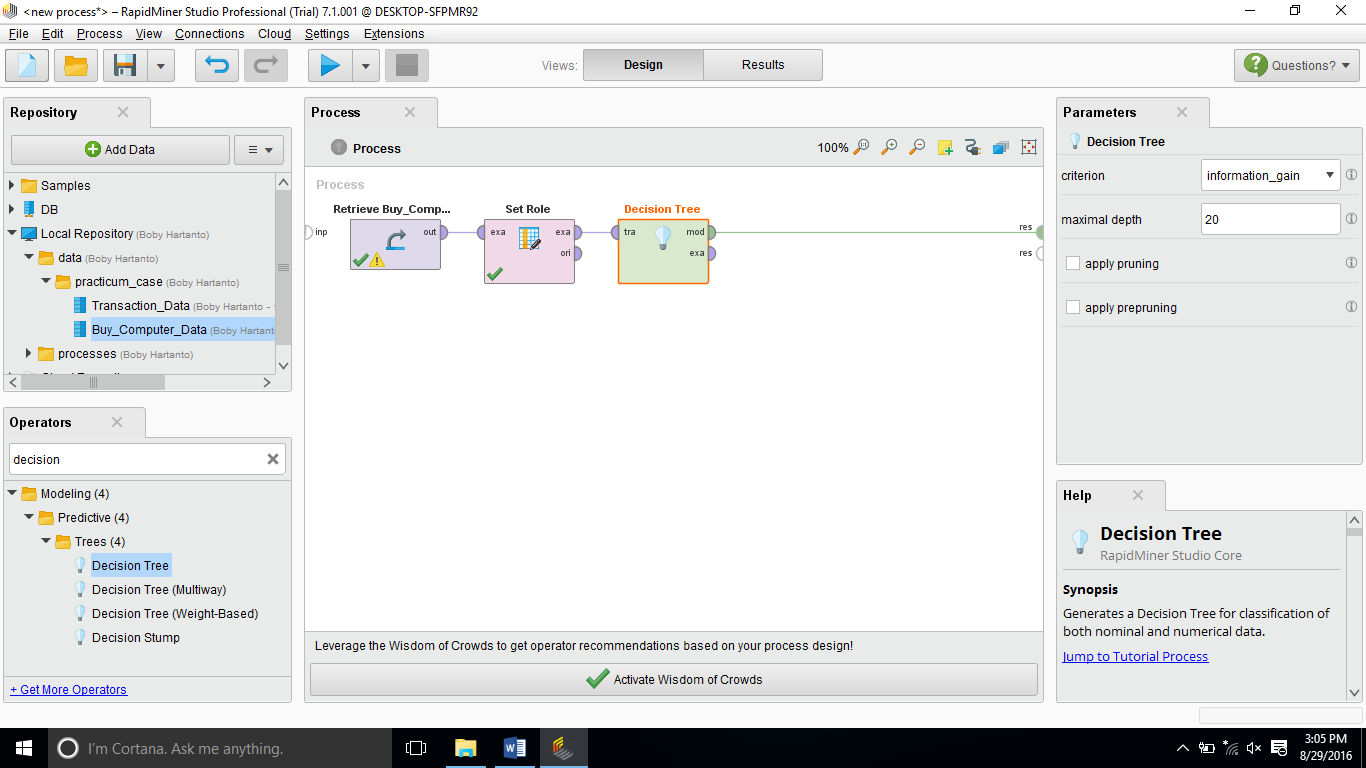
Peran yang ditentukan untuk atribut yang dipilih

Hasil:



* Step 3

Gunakan operator Decision Tree untuk membuat decision tree dari data

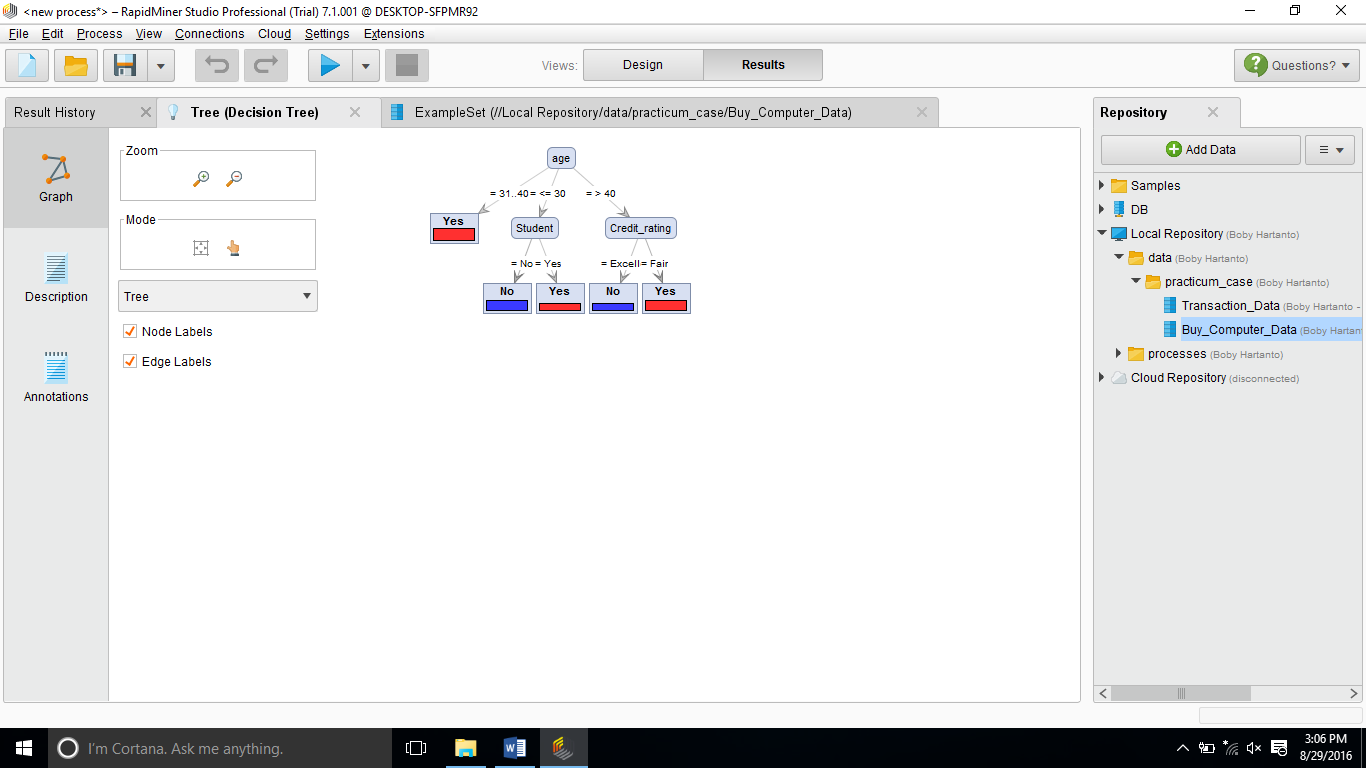


Catatan:

**Criterion**

Filter kriteria untuk menghasilkan decision tree

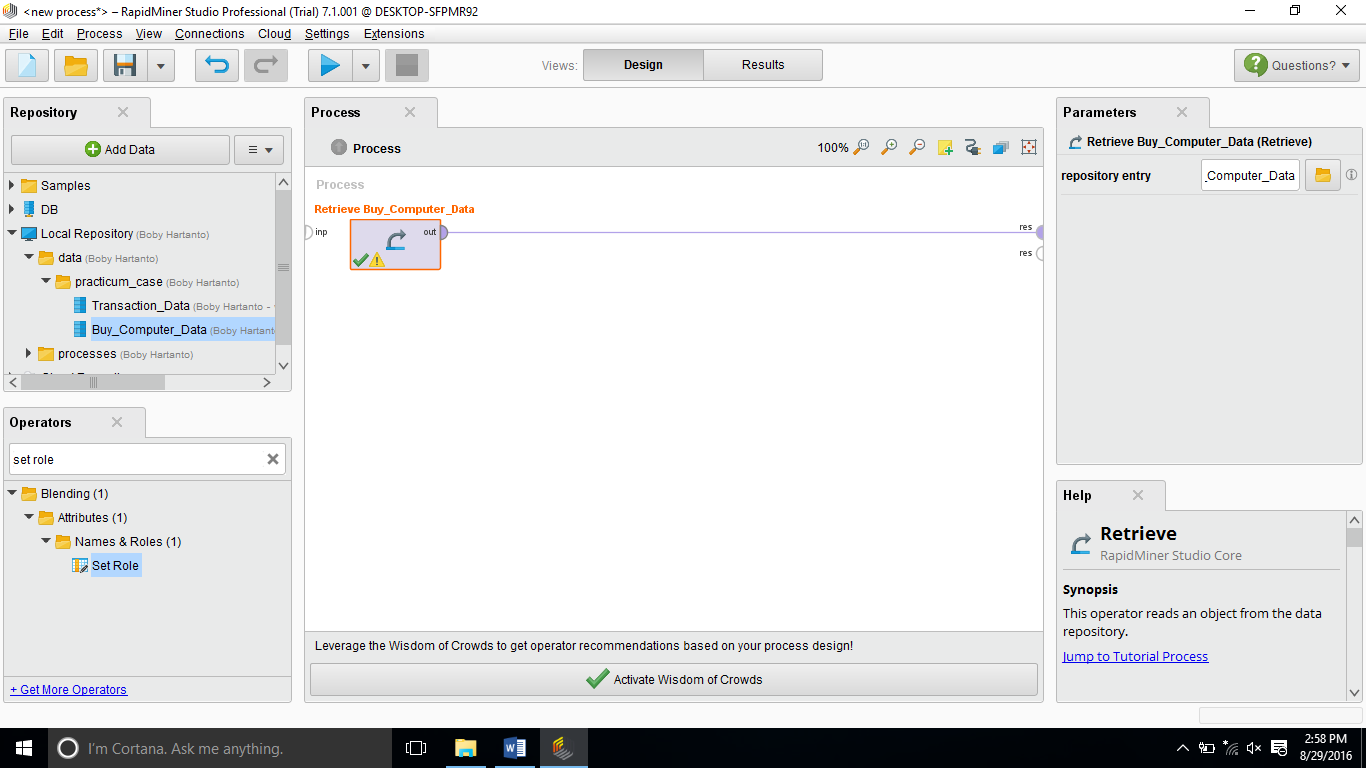
Hasil



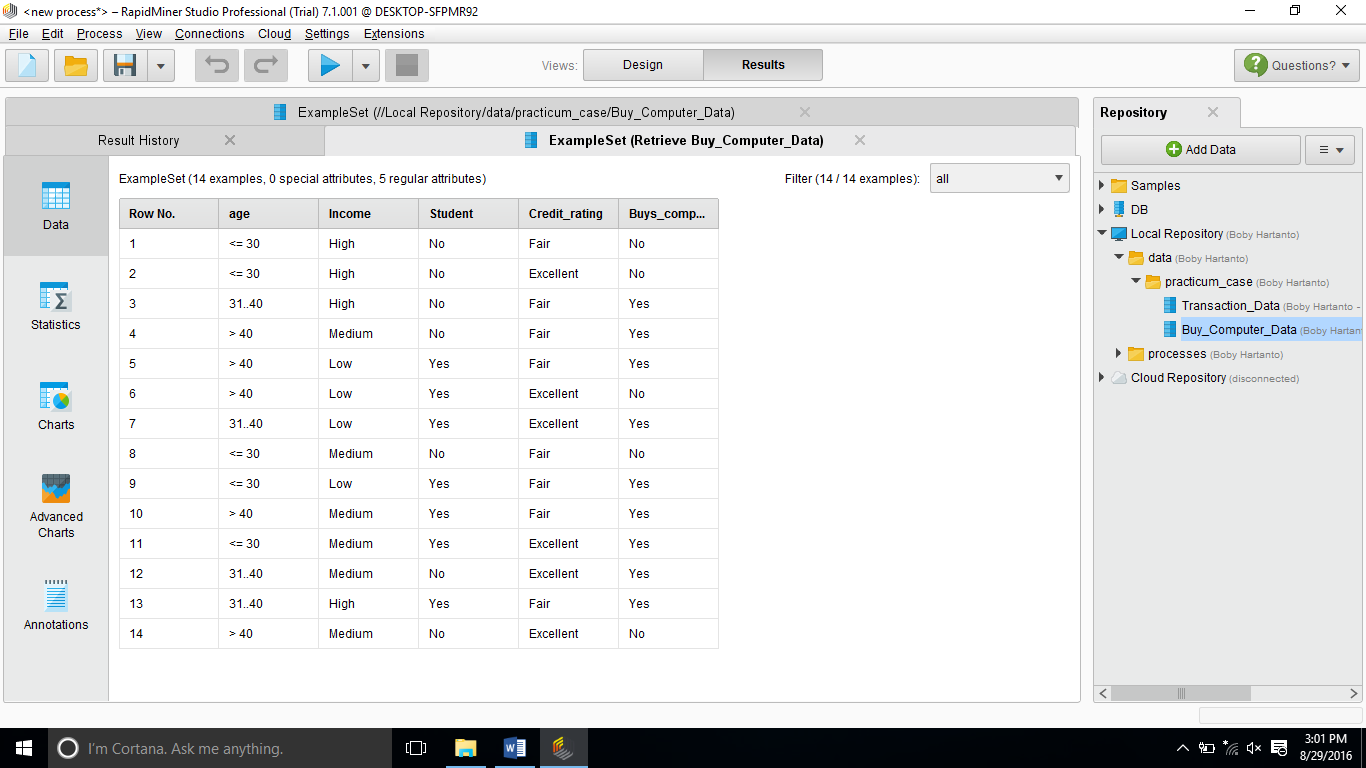
1. **K-NN**

* Langkah 1

Impor dan tarik Buy\_Computer\_Data ke dalam process window

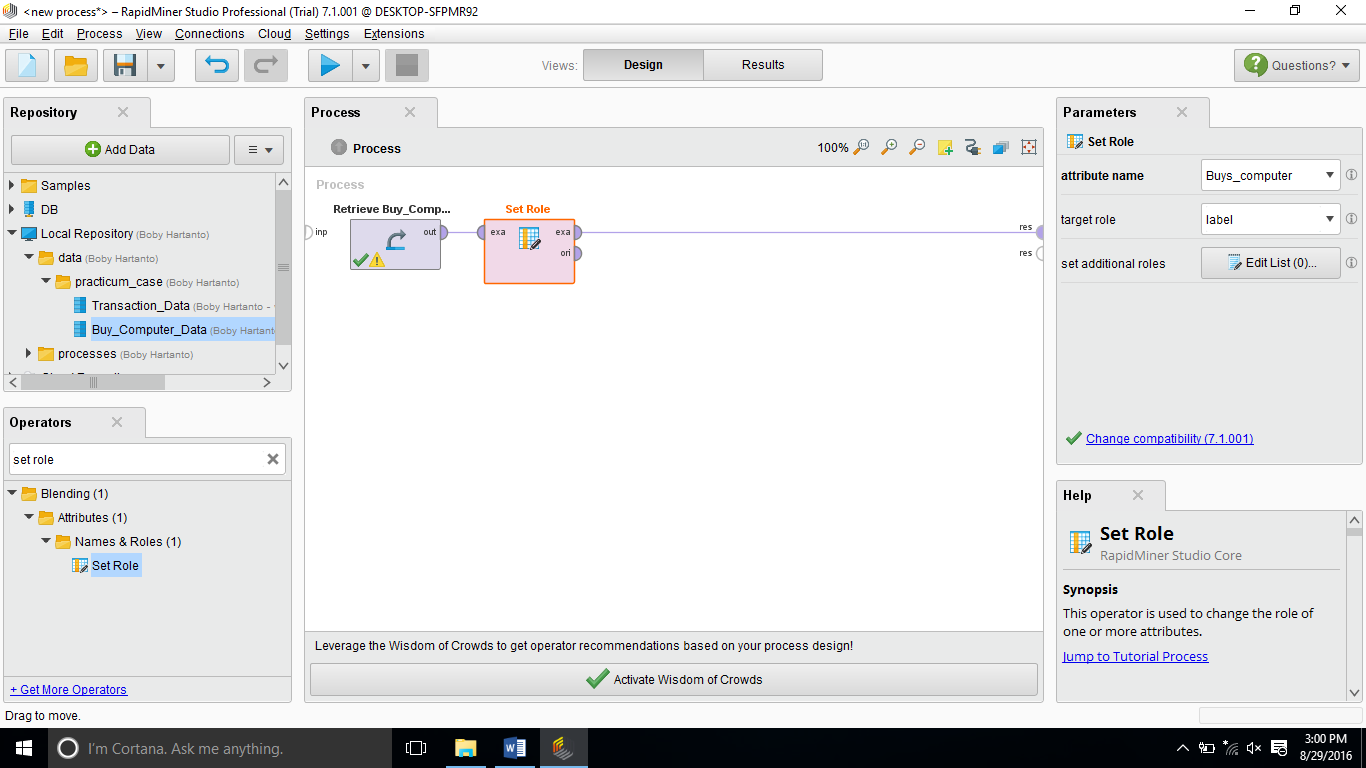


Hasil



* Step 2

Gunakan operator set role untuk mengubah peran kolom Buys\_Computer menjadi label



Catatan:

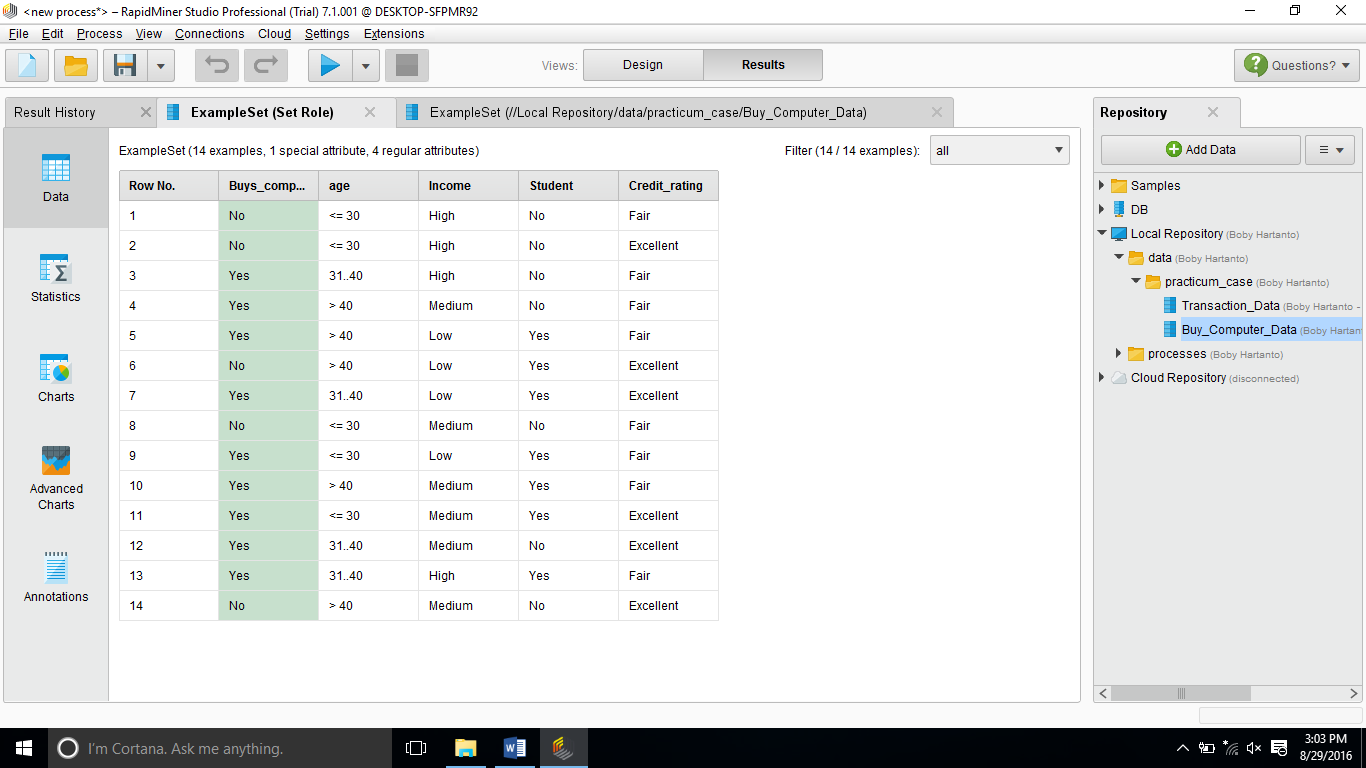
**Attribute Name**

Nama atribut yang perannya harus diubah

**Target role**

Peran yang ditentukan untuk atribut yang dipiih

Hasil



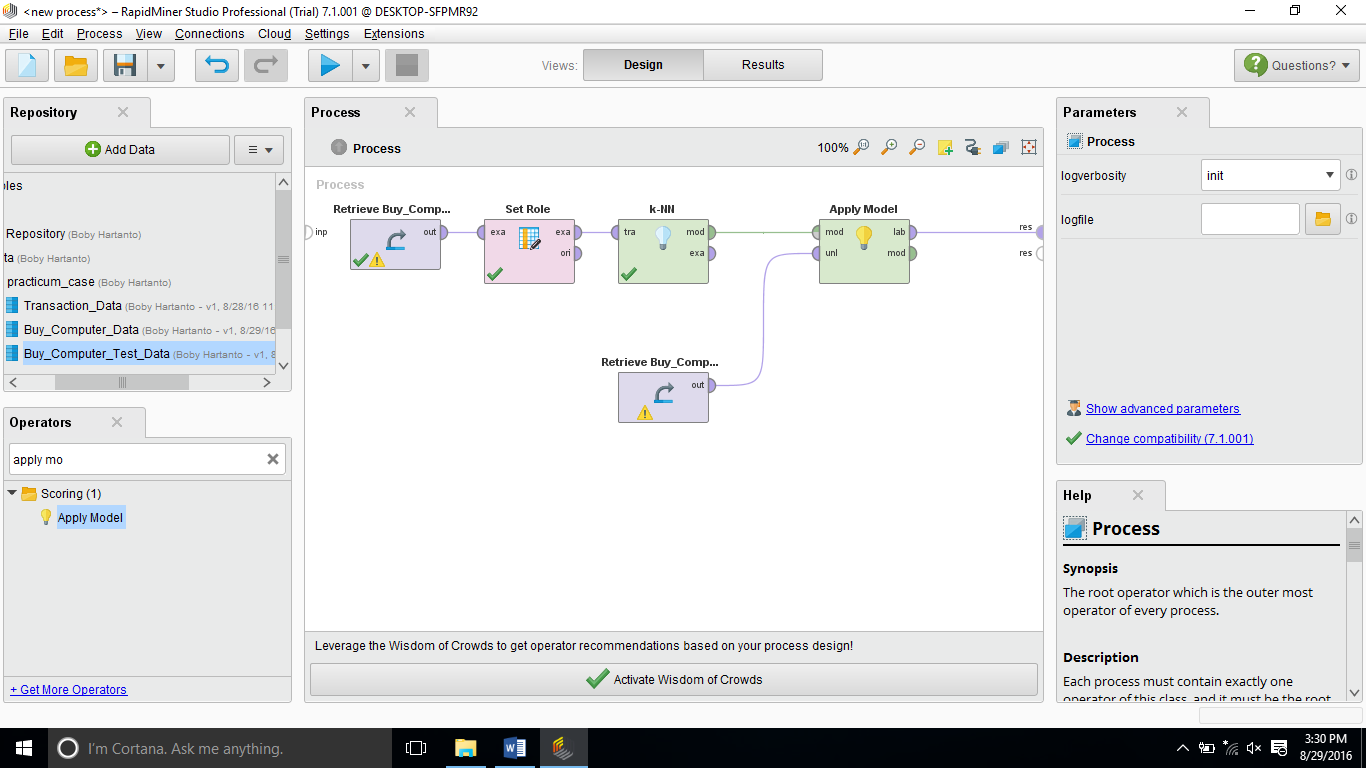
* Step 3

**K**

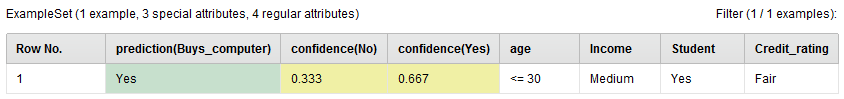
Jumlah tetangga terdekat

* Step 4

Impor data yang tidak berlabel dan mengklasifikasikan data yang menggunakan operator apply model



Hasil



Hasilnya menunjukkan prediksi Buys\_computer berdasarkan **highest confidence** (probabilitas setiap kelas yang dihitung dalam K-NN) antara setiap kelas.

Dalam contoh ini:

Confidence (No) = 0.333

Confidence (Yes) = 0.667

Karena nilai **highest confidence** adalah nilai **confidence** kelas **Buys\_computer** **=** **“Yes”**, data yang ingin diklasifikasikan diprediksi ke kelas **Buys\_computer = “Yes”**.