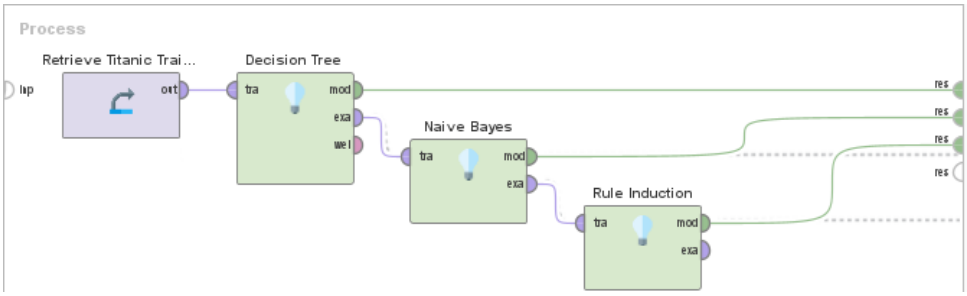
UT2_PD2

Property	COMPLETED
Date	@September 5, 2023
# UT	2

Modeling



La Modelización Predictiva es un conjunto de técnicas de aprendizaje automático que buscan patrones en conjuntos de datos voluminosos y utilizan esos patrones para crear predicciones en nuevas situaciones. Estas predicciones pueden ser categóricas (lo cual se denomina aprendizaje de clasificación) o numéricas (aprendizaje de regresión).

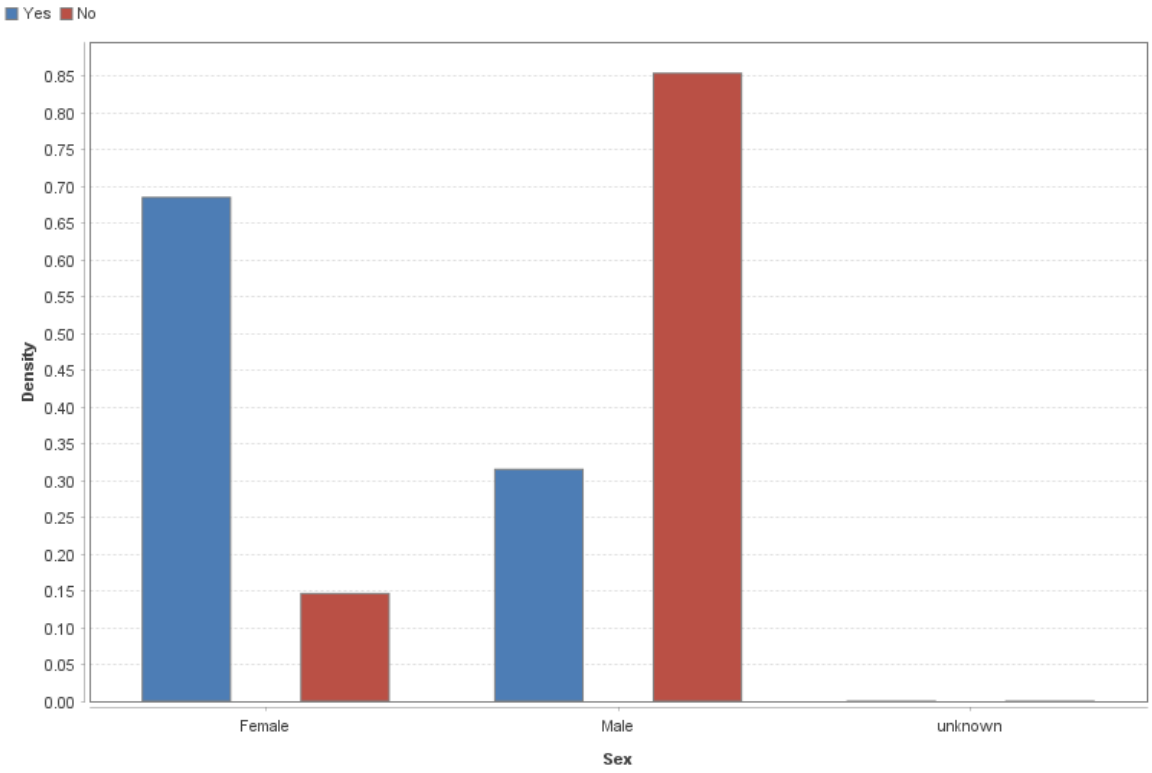
En relación al conjunto de datos del Titanic, se van a emplear tres algoritmos de clasificación diferentes:

1. Árbol de Decisión:
   El árbol de decisión es un algoritmo que modela las decisiones y sus posibles consecuencias en forma de un árbol. Cada nodo en el árbol representa una característica del conjunto de datos y cada rama representa una decisión basada en esa característica. El algoritmo divide recursivamente el conjunto de datos en subconjuntos más pequeños según los valores de las características, con el objetivo de clasificar las instancias en categorías específicas en función de las divisiones realizadas en el árbol.
2. Naive Bayes (Bayes Ingenuo): Naive Bayes es un clasificador de alto sesgo y baja varianza, y puede construir un buen modelo incluso con un conjunto de datos pequeño. Es fácil de usar y poco costoso desde el punto de vista computacional. Los casos de uso más habituales son la categorización de textos, la detección de spam, el análisis de sentimientos y los sistemas de recomendación.
   El supuesto fundamental de Naive Bayes es que, dado el valor de la etiqueta (la clase), el valor de cualquier atributo es independiente del valor de cualquier otro atributo. En sentido estricto, esta suposición rara vez es cierta (¡es "ingenua"!), pero la experiencia demuestra que el clasificador Naive Bayes suele funcionar bien. La suposición de independencia simplifica enormemente los cálculos necesarios para construir el modelo de probabilidad de Naive Bayes.
   Para completar el modelo de probabilidad, es necesario hacer alguna suposición sobre las distribuciones de probabilidad condicionales para los Atributos individuales, dada la clase. Este Operador utiliza densidades de probabilidad gaussianas para modelar los datos de Atributos.
3. Rule Induction (Inducción de Reglas):
   La inducción de reglas es un enfoque que busca identificar reglas if-then a partir de los datos de entrenamiento. Estas reglas representan patrones que relacionan ciertas condiciones con decisiones o acciones específicas. El objetivo es crear un conjunto de reglas que pueda utilizarse para predecir la pertenencia de una instancia a una categoría en función de sus características. A menudo, estos algoritmos generan conjuntos de reglas interpretables que pueden proporcionar información valiosa sobre cómo se toman las decisiones en el conjunto de datos.

Al ejecutar el Modelo encontramos los siguientes resultados:

```
RuleModel
if Sex = Male and Passenger Fare ≤ 26.269 then No  (57 / 367)
if Sex = Female and Passenger Class = First then Yes  (97 / 4)
if Sex = Male and Passenger Fare > 31.137 then No  (33 / 90)
if Passenger Class = Second and Age ≤ 28.500 then Yes  (36 / 4)
if Passenger Fare ≤ 24.808 and Passenger Fare > 15.373 and Age > 29.441 then Yes  (18 / 3)
if Passenger Fare ≤ 14.281 then Yes  (68 / 40)
if Passenger Class = Third and Passenger Fare > 23.746 then No  (1 / 23)
if Passenger Class = Second and Passenger Fare > 30.375 then Yes  (4 / 0)
if No of Parents or Children on Board ≤ 0.500 and Age ≤ 30.441 and Passenger Fare ≤ 28.710 and Age > 28.500 then No  (1 / 8)
if Age ≤ 54 then Yes  (33 / 22)
if Age ≤ 71 then No  (0 / 6)
else Yes  (0 / 0)

correct: 750 out of 915 training examples.
```

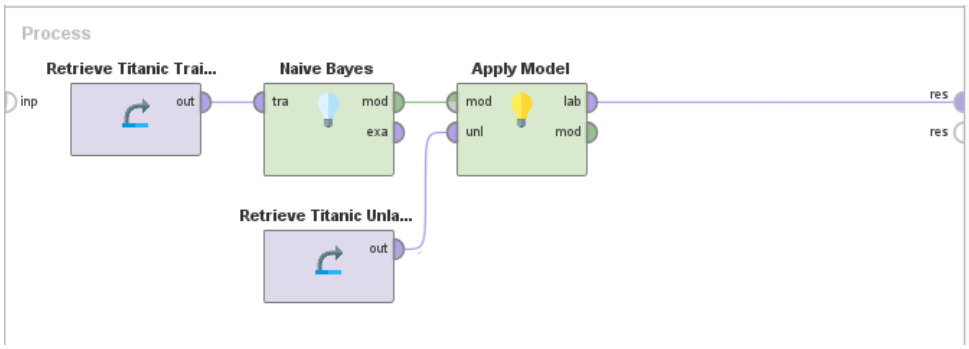
```
SimpleDistribution
Distribution model for label attribute Survived


Class Yes (0.381)
6 distributions

Class No (0.619)
6 distributions
```




```
Tree
Sex = Female
|   No of Parents or Children on Board > 4.500: No {Yes=0, No=4}
|   No of Parents or Children on Board ≤ 4.500
|   |   No of Siblings or Spouses on Board > 4.500: No {Yes=0, No=2}
|   |   No of Siblings or Spouses on Board ≤ 4.500
|   |   |   Passenger Fare > 35.562: Yes {Yes=101, No=3}
|   |   |   Passenger Fare ≤ 35.562
|   |   |   |   No of Parents or Children on Board > 3.500: No {Yes=0, No=3}
|   |   |   |   No of Parents or Children on Board ≤ 3.500
|   |   |   |   |   Passenger Fare > 33.688: No {Yes=0, No=2}
|   |   |   |   |   Passenger Fare ≤ 33.688
```

```
|   |   |   |   |   |   No of Siblings or Spouses on Board > 2.500
|   |   |   |   |   |   |   No of Parents or Children on Board > 0.500: No {Yes=2, No=11}
|   |   |   |   |   |   |   No of Parents or Children on Board ≤ 0.500: Yes {Yes=2, No=0}
|   |   |   |   |   |   No of Siblings or Spouses on Board ≤ 2.500: Yes {Yes=134, No=58}
Sex = Male
|   No of Siblings or Spouses on Board > 4.500: No {Yes=0, No=7}
|   No of Siblings or Spouses on Board ≤ 4.500
|   |   No of Parents or Children on Board > 2.500: No {Yes=0, No=6}
|   |   No of Parents or Children on Board ≤ 2.500
|   |   |   No of Parents or Children on Board > 1.500
|   |   |   |   Age > 3.500
|   |   |   |   |   Passenger Fare > 61.719
|   |   |   |   |   |   Age > 18
|   |   |   |   |   |   |   Age > 33: Yes {Yes=1, No=1}
|   |   |   |   |   |   |   Age ≤ 33: No {Yes=0, No=2}
|   |   |   |   |   |   Age ≤ 18: Yes {Yes=4, No=0}
|   |   |   |   |   Passenger Fare ≤ 61.719: No {Yes=1, No=10}
|   |   |   |   Age ≤ 3.500: Yes {Yes=3, No=0}
|   |   |   No of Parents or Children on Board ≤ 1.500
|   |   |   |   No of Siblings or Spouses on Board > 3.500: No {Yes=0, No=8}
|   |   |   |   No of Siblings or Spouses on Board ≤ 3.500
|   |   |   |   |   No of Parents or Children on Board > 0.500
|   |   |   |   |   |   Age > 5
|   |   |   |   |   |   |   Age > 62.500: No {Yes=0, No=2}
|   |   |   |   |   |   |   Age ≤ 62.500
|   |   |   |   |   |   |   |   Age > 12: No {Yes=11, No=31}
|   |   |   |   |   |   |   |   Age ≤ 12: Yes {Yes=3, No=2}
|   |   |   |   |   |   Age ≤ 5: Yes {Yes=7, No=1}
|   |   |   |   |   No of Parents or Children on Board ≤ 0.500
|   |   |   |   |   |   Passenger Fare > 26.144
|   |   |   |   |   |   |   Passenger Fare > 26.469
|   |   |   |   |   |   |   |   Passenger Class = First: No {Yes=31, No=64}
|   |   |   |   |   |   |   |   Passenger Class = Second: No {Yes=0, No=6}
|   |   |   |   |   |   |   |   Passenger Class = Third: Yes {Yes=4, No=1}
|   |   |   |   |   |   |   Passenger Fare ≤ 26.469: Yes {Yes=3, No=0}
|   |   |   |   |   |   Passenger Fare ≤ 26.144: No {Yes=42, No=343}
```

Scoring



We will use the Naïve Bayes method to predict the "Survived" class (yes / no) of each passenger and find their respective confidences.

> Using a model to generate predictions for new data points is called *Scoring*.

Apply Model: The operator takes unlabeled data as an input, applies the model you connected to the "mod" port, and outputs a data set with a label: the predictions made by the model.

Tras ejuctar el modelo encontramos que:

The result is the original unlabeled data with a column for the predicted class (yes / no) of "Survived" and two additional columns for the confidences of the two different classes (yes / no) of "Survived". For example, the first row of the data the prediction is "yes" with 98.7% confidence and "no" with 1,3% confidence.

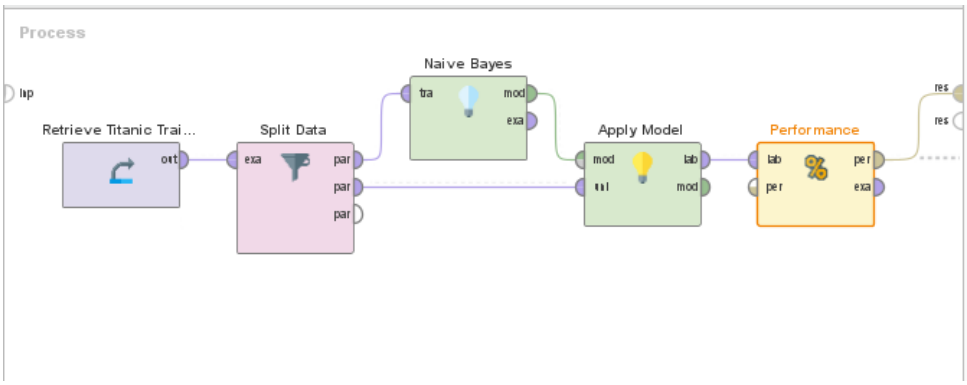
Open in Turbo Prep Auto Model Filter (392 / 392 examples): all

Row No.	prediction(S...	confidence(...	confidence(...	Age	Passenger ...	Sex	No of Sibling...	No of Parent...	Passenger F...
1	Yes	0.987	0.013	0.917	First	Male	1	2	151.550
2	Yes	0.714	0.286	53	First	Female	2	0	51.479
3	Yes	0.542	0.458	71	First	Male	0	0	49.504
4	Yes	1.000	0.000	47	First	Male	1	0	227.525
5	Yes	0.922	0.078	24	First	Female	0	0	69.300
6	Yes	0.891	0.109	47	First	Female	1	1	52.554
7	No	0.296	0.704	25	First	Male	0	0	26
8	Yes	1.000	0.000	45	First	Female	0	0	262.375
9	Yes	0.839	0.161	44	First	Female	0	0	27.721
10	Yes	0.996	0.004	41	First	Female	0	0	134.500
11	No	0.291	0.709	45	First	Male	0	0	26.550
12	Yes	1	0	58	First	Female	0	1	512.329
13	No	0.276	0.724	33	First	Male	0	0	5
14	Yes	0.988	0.012	14	First	Female	1	2	120
15	Yes	0.836	0.164	29.881	First	Female	0	0	27.721

ExampleSet (392 examples, 3 special attributes, 6 regular attributes)

Test Splits and Validation

El objetivo ahora es separar el dataset en dos Training y Test. En este caso se aplico el siguiente modelo:



Lo primero que se realiza es utilizar el operador Split Data el cual:

> Split Data takes an example set and divides it into the partitions you have defined. In this case, we will get two partitions with 70% of the data in one and 30% of the data in the other. Both sets are still labeled. The 70% partition will become our training set we build our model on. The remaining 30% will become our test set against which we can compare our model's predictions. This 70/30-ratio between training and testing is actually quite a popular and effective value.

Table View Plot View

accuracy: 80.36%

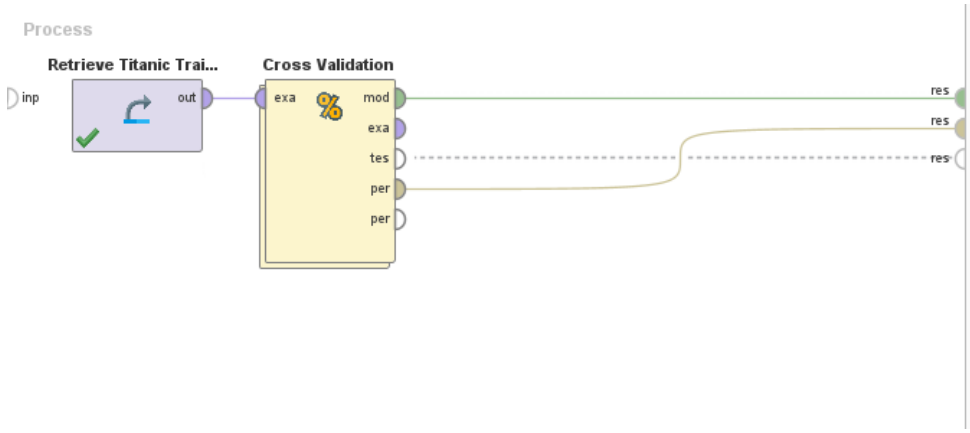
	true Yes	true No	class precision
pred. Yes	76	25	75.25%
pred. No	29	145	83.33%
class recall	72.38%	85.29%	

The first result you see is the test set with the label and the predictions. The second result is the performance of the model on the test set. You can select the different performance measurements ("criterion") on the left side of the screen. The accuracy is 80.36% and tells you how accurate the model is overall. The confusion matrix shows you the different kind of errors. For example, 29 cases have been predicted "no" when they were actually "yes." The accuracy is the sum of all the numbers on the diagonal divided by the sum of all numbers! The larger the numbers on the diagonal, the better the performance of our model.

Use Performance operator to calculate how well the model does. Work on the challenges below and proceed to the next tutorial.

Cross Validation

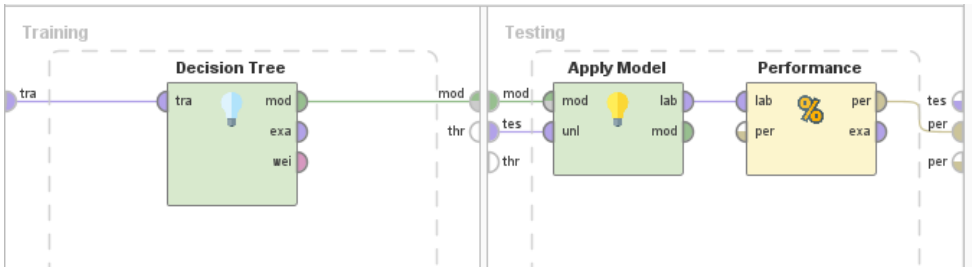
Cross Validation is a technique to make sure each data point is used as often for training as it is for testing which avoids this problem.



Cross validation divides the example set into equal parts and rotates through all parts, always using one for testing and all others for training the model. At the end, the average of all testing accuracies is delivered as result. This is a great way to compute the accuracy of models and should become your standard estimation approach whenever the additional computation efforts are feasible.

By default this splits the data into 10 different parts, so we call this a 10-fold cross validation.

El operador Cross Validation puede tener subprocesos dentro tal como se ve:



Resultados:

accuracy: 80.35% +/- 4.69% (micro average: 80.35%)

	true Yes	true No	class precision
pred. Yes	253	84	75.07%
pred. No	96	483	83.42%
class recall	72.49%	85.19%	

+/- stands for DEVEST

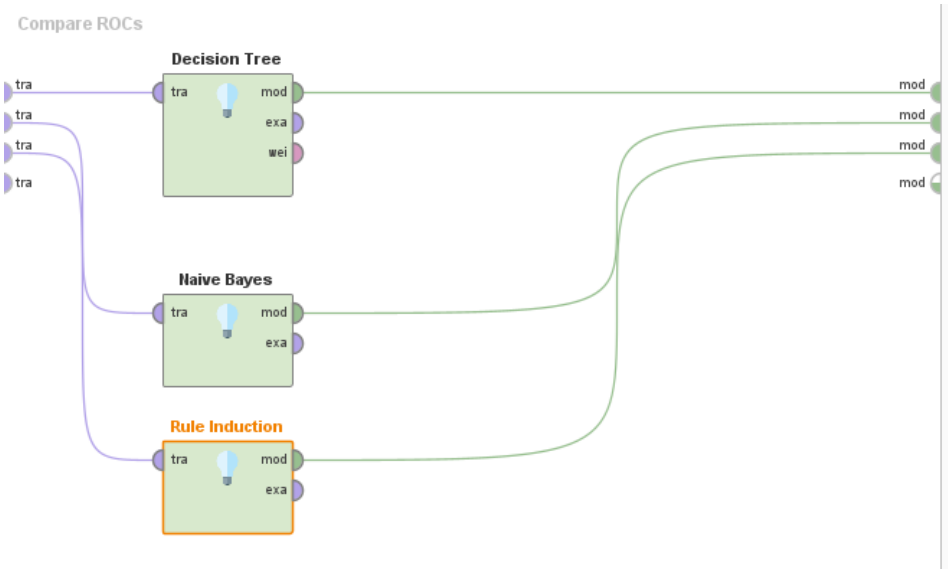
The standard deviation gives us an idea of how robust the model is: the smaller the standard deviation, the less dependent the model performance is on the test data set.

Visual Model Comparison

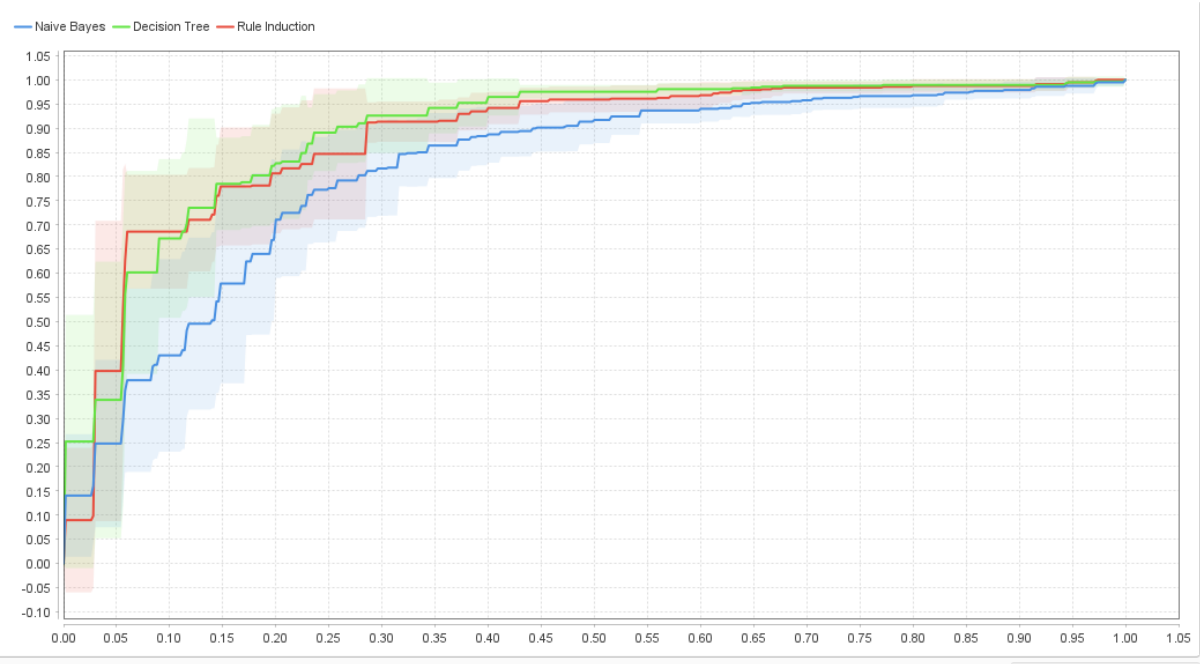
The Receiver Operating Characteristics (ROC) curve shows how well a binary machine learning model works. It shows the True Positive Rate (TPR) against the False Positive Rate (FPR) for different confidence thresholds of the model (learn more here).

> ROC curves are a well known way to visualize the performance of models. Don't worry if you do not know the math behind them: Just remember that the curves of better models move to the top left. A perfect model produces a line that goes straight up (vertically) and then straight to the right (horizontally).

Lo que se realizo fue: Se cargo el dataset del titanic para training al cual se le conecto el operador ROC, este operador nos permite definir un subproceso. En este caso agregamos tres algoritmos de clasificacion diferentes:



Tras ejecutar el modelo se obtuvo el siguiente resultado:



The chart shows all three models curving towards the top-left corner, so we know they are all more effective than random guessing. In this case, Naïve-Bayes is the farthest away from the top-left corner, meaning it performs worst in this case