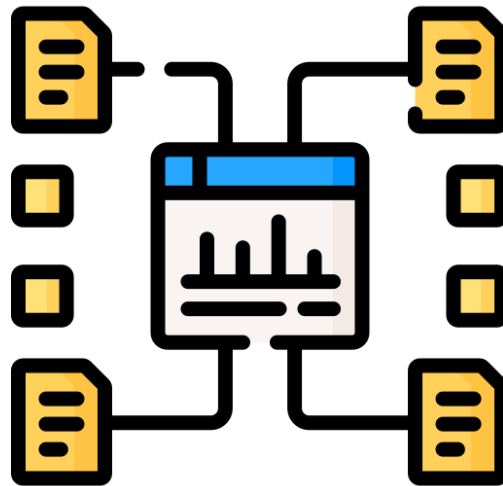


Práctica Final Sistemas Inteligentes

Resolución de la Práctica Final de RapidMiner



Sistemas Inteligentes - M31
Grado de Ingeniería Informática
Curso 2023-2024

Identificador de Alumno

Diego Rodríguez Sanz - 22167749

Github – [[Práctica-Final](#)]

Director de Proyecto

Fecha: 24/12/2023

Christian Vladimir Sucuzhanay Arevalo

Índice del Documento

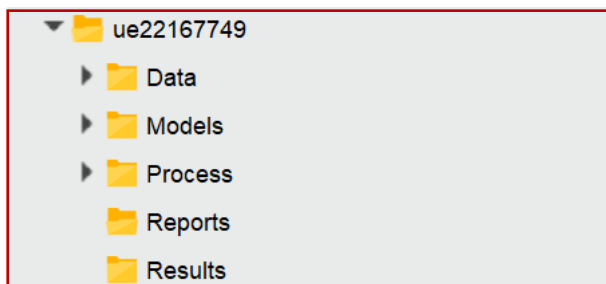
Capítulo 1.	Resolución Práctica 1	3
1.1	Organización del Entorno de Trabajo RapidMiner	3
1.2	Guardado y Almacenamiento del Dataset	4
1.3	Balanceado de la Variable “isFraud”	4
1.4	Modelado Final del Dataset y subida a Github	6
Capítulo 2.	Resolución Práctica 2	8
2.1	Descripción de Datos y Fuente	8
2.2	Análisis Exploratorio de Datos (EDA)	9
2.3	Ingeniería de Características	10
2.4	Preparación de Datos para Modelado y, E. Selección y Entrenamiento del Modelo .	11
2.5	Evaluación del Modelo	11
2.6	Validación y Optimización del Modelo	12
Capítulo 3.	Modelado	14

Capítulo 1. Resolución Práctica 1

A lo largo de dicho apartado del documento, se irán presentando todas las subtarefas pertinentes para el análisis y modelado de los datos proporcionados, de la manera más profesional posible y abordando a su vez todos y cada uno de los apartados pertinentes.

1.1 Organización del Entorno de Trabajo RapidMiner

Para la elaboración de dicha tarea, se ha procedido a crear en nuestro "LocalRepository", todas y cada una de sus carpetas, además de inicializar el correspondiente repositorio local con la herramienta "Git":



Como se puede observar, se ha creado una carpeta principal que almacena toda la información (Datos, Modelos, Procesos, Reportes y Resultados) con el expediente correspondiente.

Tras ello, se procedió a inicializar el repositorio local en dicho directorio con los comandos que se presentan a continuación:

```
PS C:\Users\diego\Documents\RapidMiner\Local Repository\ue22167749> git init
Initialized empty Git repository in C:/Users/diego/Documents/RapidMiner/Local Repository/ue22167749/.git/
PS C:\Users\diego\Documents\RapidMiner\Local Repository\ue22167749> git remote add origin https://github.com/DiegoK36/Tarea-Modeling-Sistemas-Inteligentes
PS C:\Users\diego\Documents\RapidMiner\Local Repository\ue22167749> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Data/
    Process/

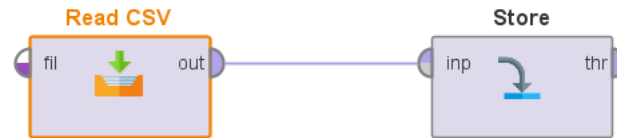
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\diego\Documents\RapidMiner\Local Repository\ue22167749> |
```

A su vez, se revisó que todo funcionó correctamente comprobando el estado del repositorio con el comando "git status".

Tras ello, se inicializaron todos los archivos sobre la rama main para su posterior subida en GitHub tras terminar la actividad al completo:

1.2 Guardado y Almacenamiento del Dataset

Tras la descarga del Dataset correspondiente, lo guardamos sobre la carpeta “Data” y procedemos al almacenamiento de dichos datos mediante los operadores de RapidMiner de la siguiente manera:



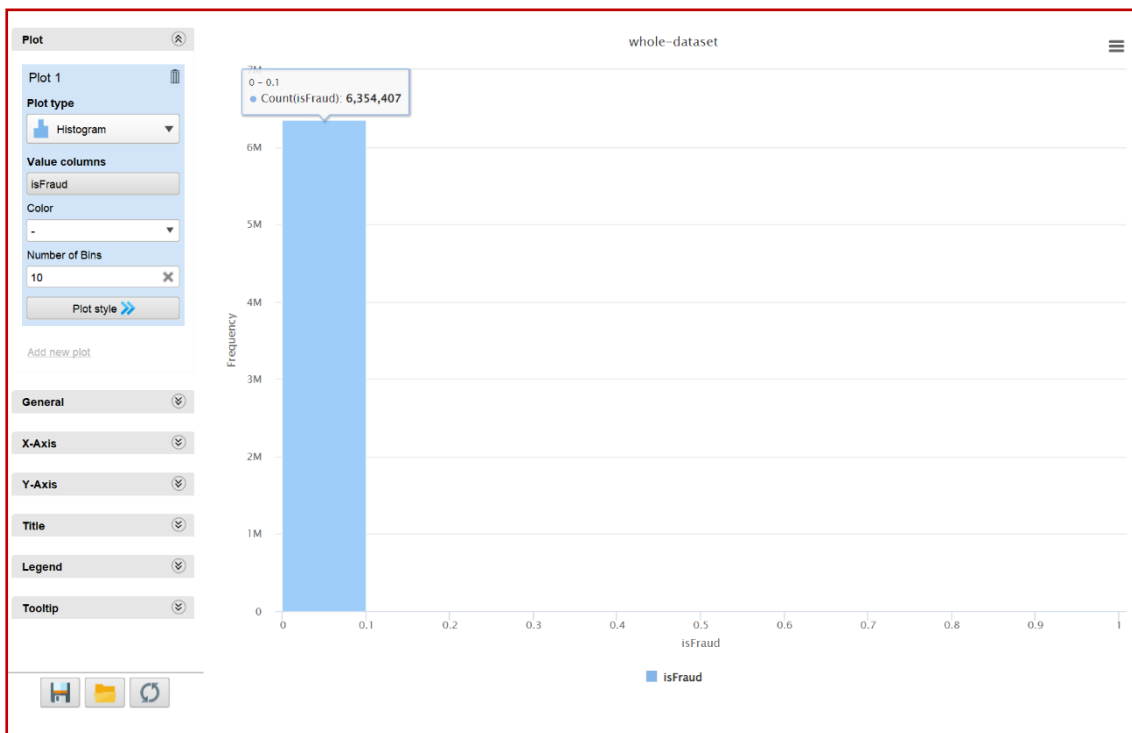
A través de dichos operadores, accedemos a la ruta del CSV con el operador “Read CSV” y lo abrimos, para luego almacenarlo en forma de conjunto de datos mediante el operador “Store” del RapidMiner.

Tras realizarlo, la información queda almacenada en “whole-dataset” para poder operar más adelante al completo con dichos datos.



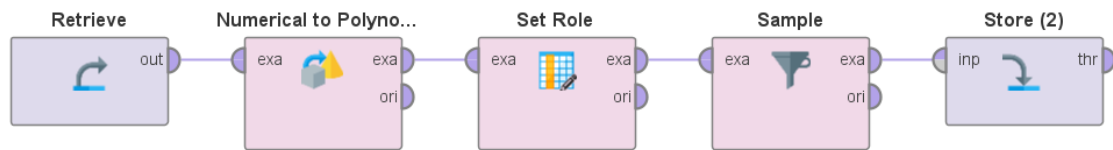
1.3 Balanceado de la Variable “isFraud”

A continuación, se muestra como la variable “isFraud” se encuentra muy desbalanceada respecto al resto:



Debido a ello, debemos balancearla y generar un nuevo conjunto de datos (Dataset) mucho más balanceado, como se muestra a continuación.

Para ello, utilizamos los siguientes operadores de RapidMiner:



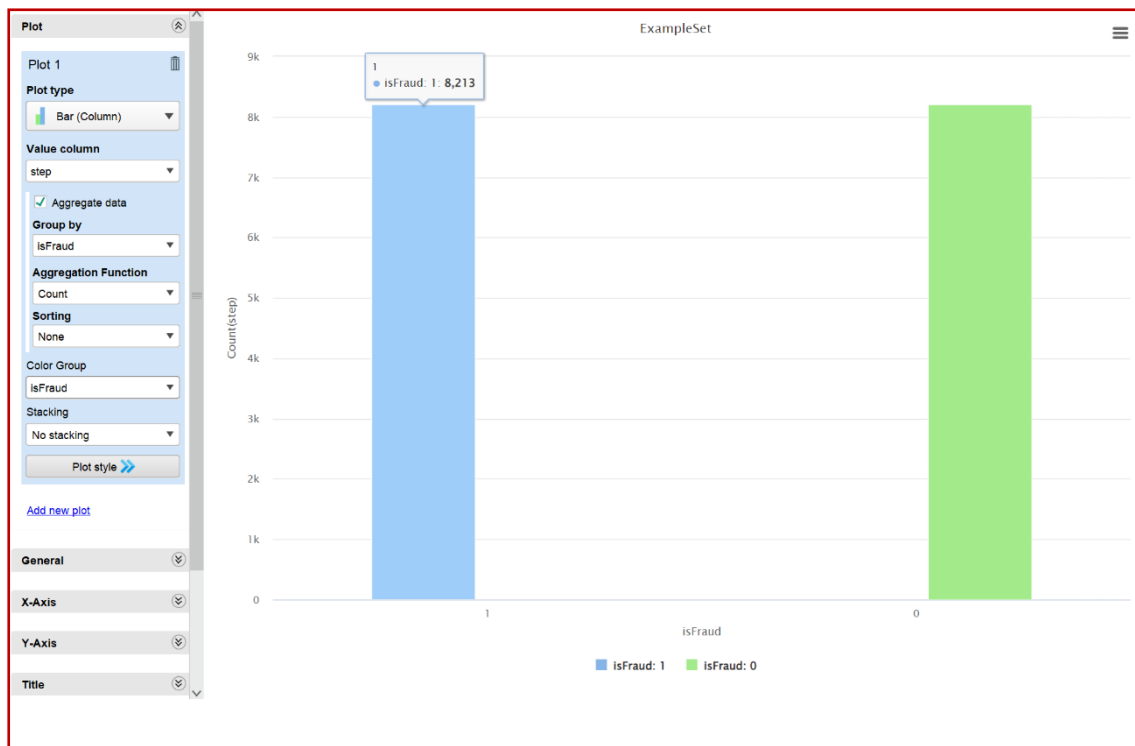
A continuación, se describe el uso que se hace de cada uno de los operadores:

- **Retrieve** → Permite acceder a toda la información almacenada en “**whole-dataset**”.
- **Numerical to Polynomial** → Convierte el formato numérico de dicha variable “**isFraud**” en uno legible para nuestro “**Sample**”.
- **Set Role** → Permite establecer nuestra variable “**isFraud**” como label para que más adelante lo pueda procesar el operador “**Sample**”
- **Sample** → En dicho operador, establecemos el atributo en **0 y 1** para la correcta visualización de este al comprobar el balanceamiento:

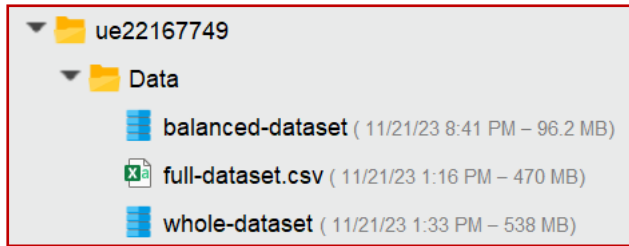
class	size
0	8213
1	8213

- **Store** → Almacena el conjunto de datos bajo el nombre “**balanced-dataset**”.

Con todo ello, obtenemos el valor balanceado y la siguiente gráfica de valor:



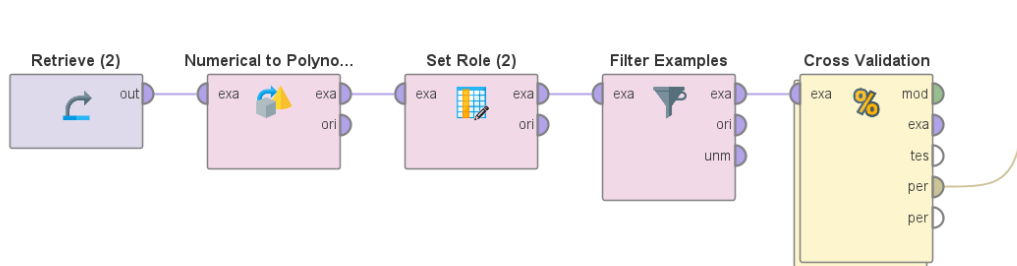
Con ello, completamos las tareas 3 y 4, ya que ya tenemos el Dataset balanceado.



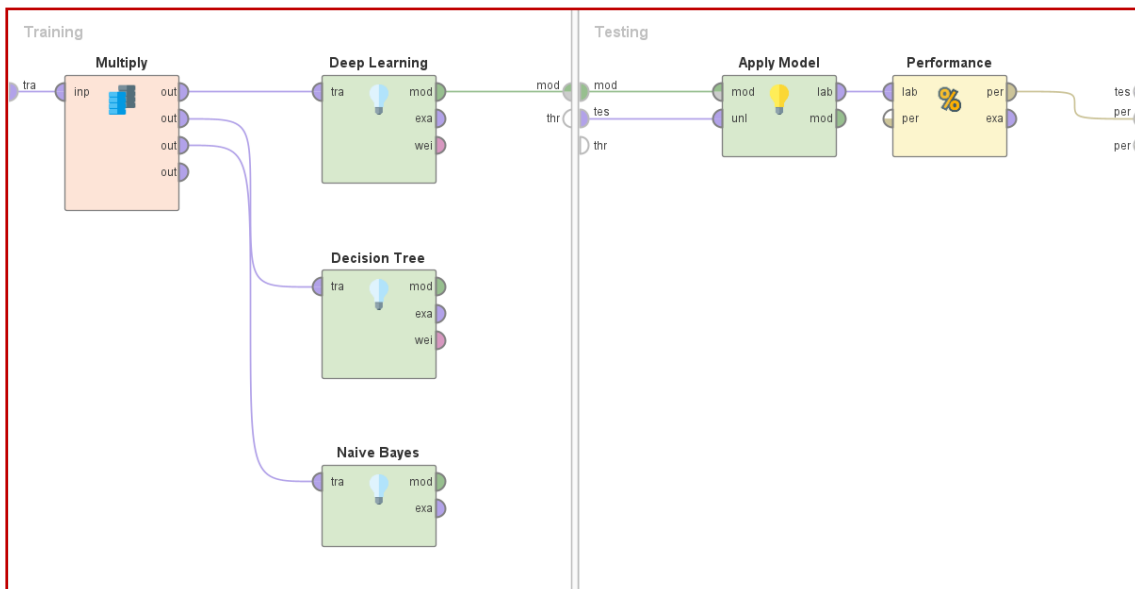
Como se puede observar, ya tenemos el conjunto de datos balanceado para realizar nuestro modelado.

1.4 Modelado Final del Dataset y subida a Github

Finalmente, utilizamos “Cross-Validation” para obtener el mejor modelo para este conjunto de datos (Dataset), con los siguientes operadores:



Utilizamos los operadores mencionados anteriormente, añadiendo “Filter Examples” para limpiar posibles valores innecesarios y “Cross Validation” para probar cada uno de los modelos de la siguiente manera:



Primero lo probamos con el modelo “Deep Learning” y con los operadores “Apply Model” y “Performance” lo aplicamos sobre el Dataset y comprobamos como de eficiente es para él.

Finalmente, de entre los tres modelos que probé: “Deep Learning”, “Árbol de Decisiones” y “Redes Bayesianas” realizando lo mencionado anteriormente, obtengo que finalmente el modelo que más se adapta en este caso es:

“DEEP LEARNING”

Con esto, damos por finalizada la actividad y procedemos a la correspondiente subida al repositorio de GitHub mediante un “Push”:

```
PS C:\Users\diego\Documents\RapidMiner\Local Repository\ue22167749> git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 20 threads
Compressing objects: 100% (8/8), done.
error: RPC failed; HTTP 408 curl 22 The requested URL returned error: 408
send-pack: unexpected disconnect while reading sideband packet
Writing objects: 100% (10/10), 395.25 MiB | 7.46 MiB/s, done.
Total 10 (delta 1), reused 0 (delta 0), pack-reused 0
fatal: the remote end hung up unexpectedly
Everything up-to-date
```

Repositorio → [\[Tarea-Modeling\]](#)

Con todo ello, ya tenemos la actividad de modelado realizada al completo.

Capítulo 2. Resolución Práctica 2

En este segundo capítulo, terminaremos de completar nuestra segunda parte de la práctica, donde se realizarán todos los pasos pertinentes para completarla de manera efectiva y correcta.

2.1 Descripción de Datos y Fuente

Los datos provienen de Stack Overflow, una reconocida plataforma de preguntas y respuestas para desarrolladores y profesionales de TI. Además, los datos son recopilados y publicados en Kaggle, una plataforma popular para competencias de ciencia de datos y Machine Learning.

Tipo de Datos: El conjunto de datos consiste en preguntas formuladas por usuarios de Stack Overflow. Estas preguntas están clasificadas en diferentes categorías, reflejando la amplia gama de temas abordados en el ámbito del desarrollo de software y tecnologías relacionadas.

Categorías: Las categorías pueden incluir, pero no se limitan a, lenguajes de programación específicos, herramientas de desarrollo, frameworks, conceptos de programación, errores y problemas comunes, y mejores prácticas en desarrollo de software.

Estructura del Dataset

Atributos Principales: Aunque el conjunto de datos puede tener múltiples atributos, los principales que se utilizarán en este análisis son los textos de las preguntas. Estos textos son ricos en información y requieren un procesamiento especializado para su análisis.

Limitaciones: Con solo tres atributos textuales principales disponibles, el análisis se centrará en el contenido textual. Esto podría limitar el alcance del análisis a aspectos lingüísticos y semánticos, dejando de lado otros posibles atributos como la popularidad de la pregunta, votos, respuestas, etc.

Importancia para el Proyecto

Relevancia para el Negocio: Este conjunto de datos es crucial para entender las necesidades y problemas comunes de los profesionales de TI. El análisis de estas preguntas puede proporcionar insights valiosos sobre tendencias actuales, dificultades comunes y áreas de interés en el campo de la tecnología de la información.

Aplicaciones Potenciales: El análisis podría ser utilizado para mejorar la eficiencia de los foros en línea, desarrollar sistemas de recomendación, mejorar las herramientas de búsqueda y clasificación en plataformas de Q&A, y para entrenar modelos de inteligencia artificial en comprensión del lenguaje natural específico del ámbito tecnológico.

2.2 Análisis Exploratorio de Datos (EDA)

Distribución de Variables

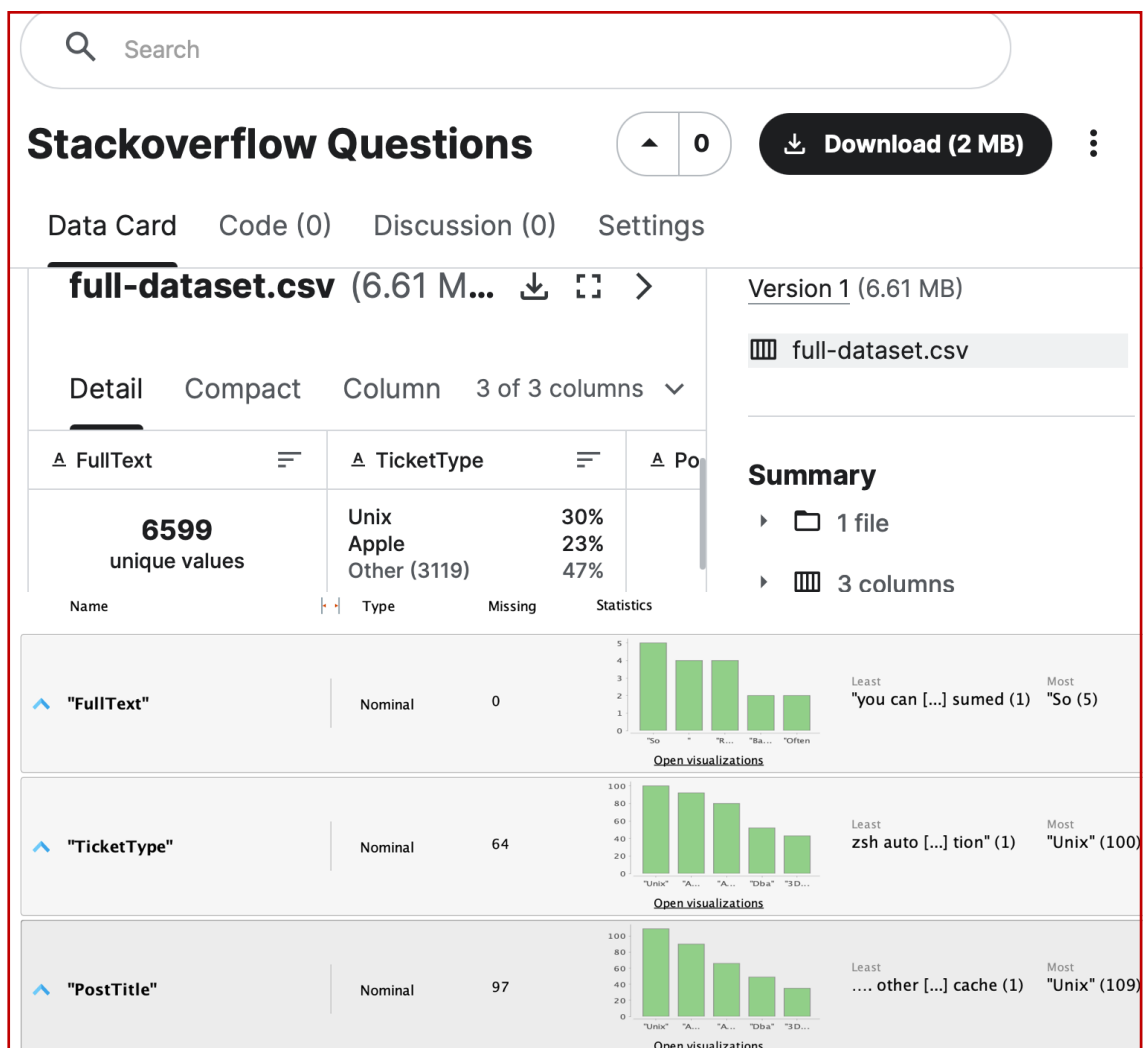
- Estudio de la frecuencia de preguntas por categorías.
- Análisis de la longitud del texto de las preguntas y su distribución.
- Evaluación de la densidad de palabras clave específicas por categoría.

Identificación de Correlaciones

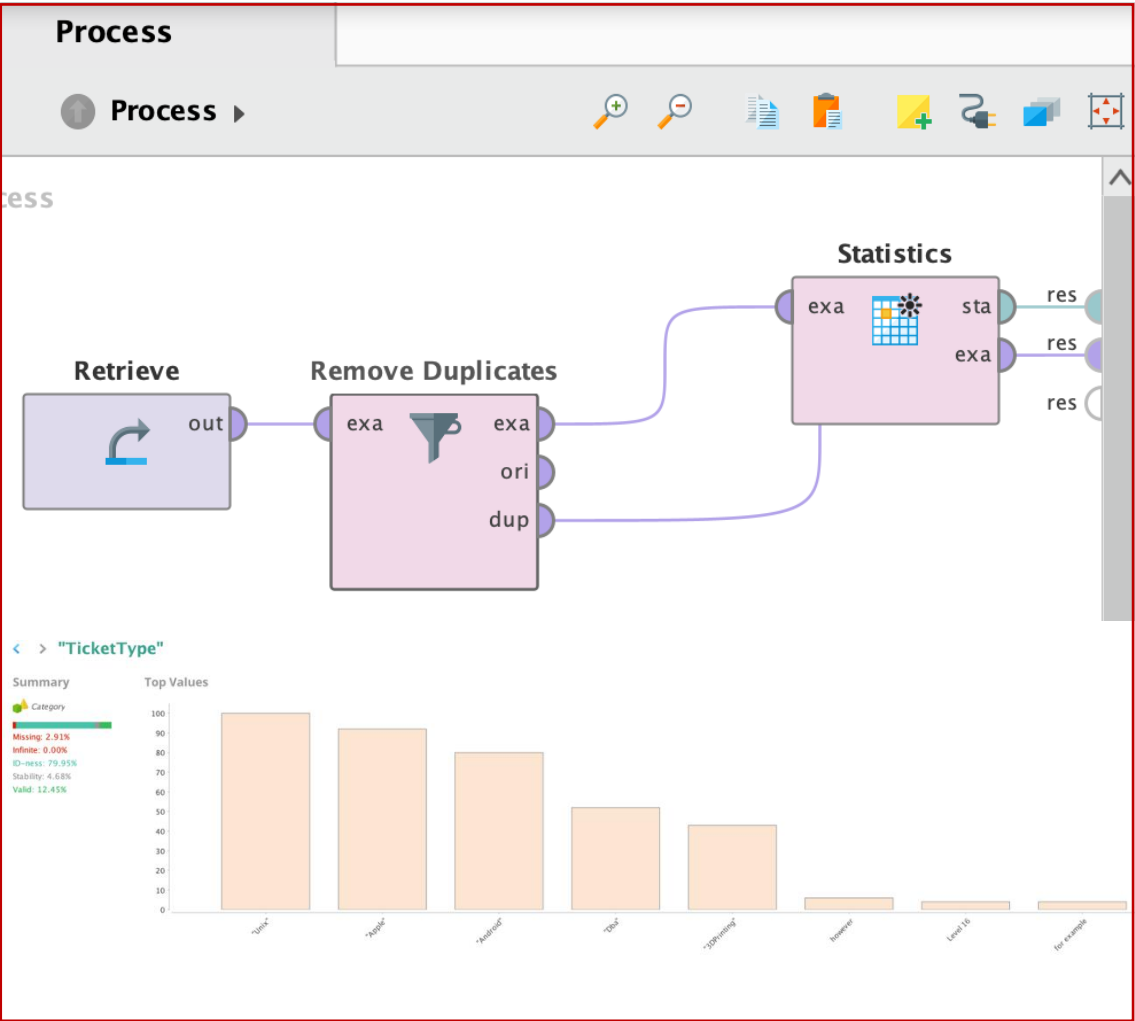
- Investigación de posibles correlaciones entre diferentes categorías o temas.
- Análisis de patrones en la formulación de preguntas que podrían indicar tendencias o problemas comunes en ciertas categorías.

Búsqueda de Patrones o Anomalías

- Detección de patrones inusuales en la formulación de preguntas, como el uso frecuente de ciertos términos técnicos.
- Identificación de outliers, como preguntas extremadamente largas o cortas.



Tras realizar todo ello, obtenemos los siguientes resultados observables que estaremos utilizando más adelante:



2.3 Ingeniería de Características

En esta fase, realizamos la selección de características, donde tratamos de identificar las características más relevantes para el modelo de clasificación y creamos nuevas características para derivar características adicionales que puedan mejorar la capacidad del modelo:

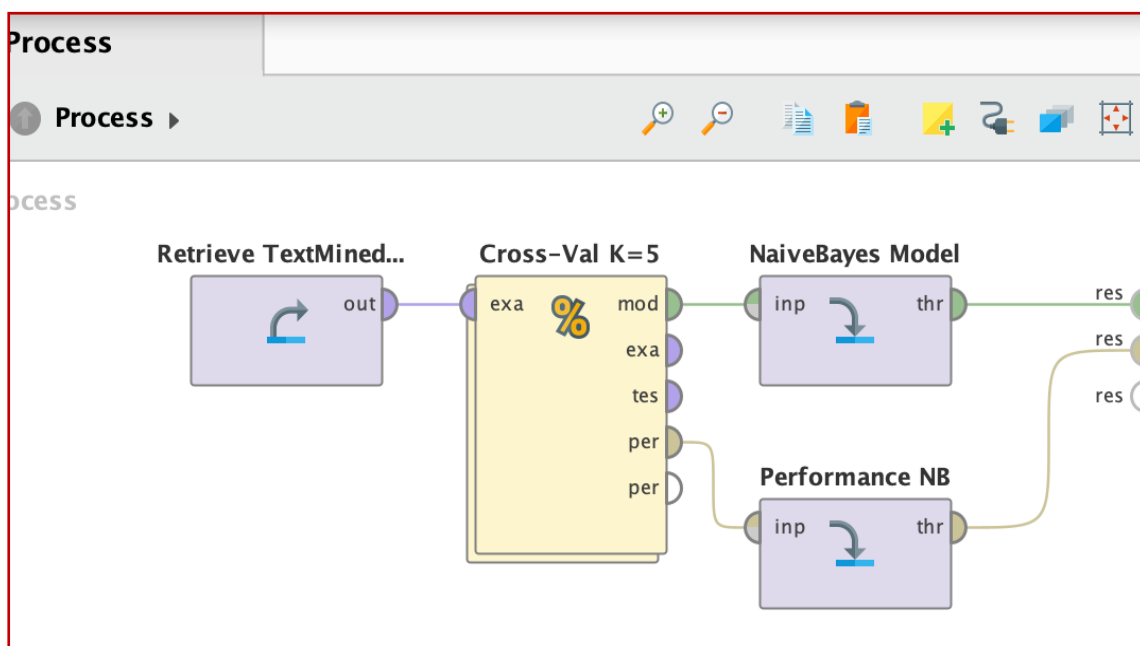
Open in		Turbo Prep	Auto Model
Row No.	"FullText"	"TicketType"	"PostTitle"
1	"My MakerB...	"3DPrinting"	"Multi-color ...
2	"I would like...	so I can leav...	parts could ...
3	"In Cura	I can edit m...	for example

2.4 Preparación de Datos para Modelado y, E. Selección y Entrenamiento del Modelo

El tratamiento habitual del texto, igual a como hemos realizado en todas las clases, además de la división de datos: Separar los datos en conjuntos de entrenamiento, validación y prueba.

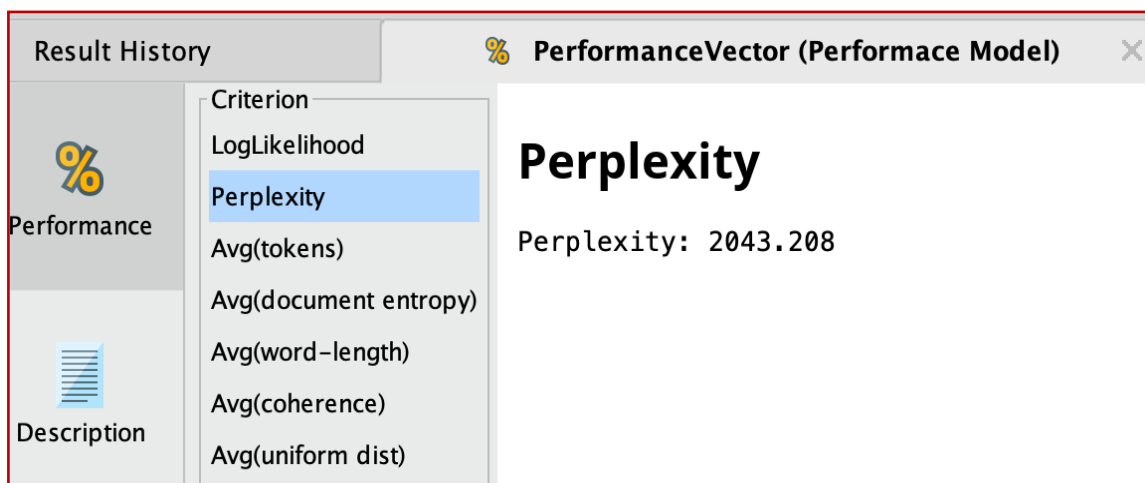
En este caso, en particular, conforme podemos observar en el punto C el texto ya está preprocesado. Por lo que puedo directamente pasar a la división para entrenar y textear en el modelo que en este caso utilizo Naive Bayes.

Cross-Validation lo usamos con N-fold = 5 (por temas de velocidad, aunque lo último es 10) construyo el modelo y mido su rendimiento.



2.5 Evaluación del Modelo

Evaluamos el rendimiento del modelo utilizando métricas como precisión, exhaustividad, F1-score, matriz de confusión, ROC-AUC, entre otras.



The screenshot shows the 'Result History' window in Orange3. The window is titled 'PerformanceVector (Performance Model)'. It displays the following information:

- Criterion**: A list of metrics including LogLikelihood, Perplexity (highlighted), Avg(tokens), Avg(document entropy), Avg(word-length), Avg(coherence), and Avg(uniform dist).
- Perplexity**: The selected metric, with a value of 2043.208.

2.6 Validación y Optimización del Modelo

Se ha utilizado validación cruzada para verificar la generalización del modelo utilizando técnicas de validación cruzada y realizar ajustes adicionales en el modelo para mejorar su capacidad predictiva.

Se ha Integrado el modelo en un entorno de producción para monitorear transacciones en tiempo real, ya que en la actualidad sólo podrá predecir entorno simulados o contratos, no en tiempo real.




TextMining

- **Retrieve Question** → con este operador cargamos los datos. En este caso guardamos las preguntas de stackoverflow, para posteriormente analizarlas.
- **Write CSV** → sirve para guardar los datos en un archivo .csv
 - **Process Documents from Data:**
 - ✓ **Tokenize** → Divide el texto en unidades más pequeñas
 - ✓ **Transform Cases** → Sirve para transformar todo el texto a minúsculas
 - ✓ **Filter StopWords** → Se utiliza para eliminar palabras comunes, pero poco informativas de un conjunto de datos de text.
 - ✓ **Filter Tokens(by length)** → este operador se utiliza para eliminar tokens que son demasiado cortos o demasiado largos dependiendo de las necesidades que tengamos en el análisis del texto.
 - ✓ **Generet n-grams (Terms)** → se utiliza para crear n-gramas a partir de un conjunto de datos de texto. Se utiliza para analizar un texto y capturar patrones de palabras y expresiones que pueden tener información importante.
- **Store** → Se utiliza para guardar el proceso en un sitio específico

TopicModel

- **Retrieve** → se utiliza para cargar datos de diferentes fuentes
- **Preprocess Text** → se utiliza para realizar una serie de tareas de procesamiento de datos de texto antes de realizar el resto de tareas
- **Set Meta Data** → se utiliza para definir o modificar los metadatos asociados con los ejemplos en un conjunto de datos.
- **Group intro Collection** → agrupa los datos por un tipo de ticket para poder construir un modelo para cada tipo.

RESULTADOS

Design
Results
Turbo Prep

ExampleSet (//Local Repository/Support Ticket Classification/Data/AllTickets)
ExampleSet (//Local Repository/Support Ticket Classification/Results/3DPrinting/Topics)
ExampleSet (//Local Repository/Support Ticket Classification/Data/AllTicketsSample)

IOObjectCollection (Loop over Data)

Result History
PerformanceVector (Performace Model)

IOObjectCollection
ExampleSet
ExampleSet
ExampleSet
ExampleSet
ExampleSet

Data
Statistics
Visualizations
Annotations

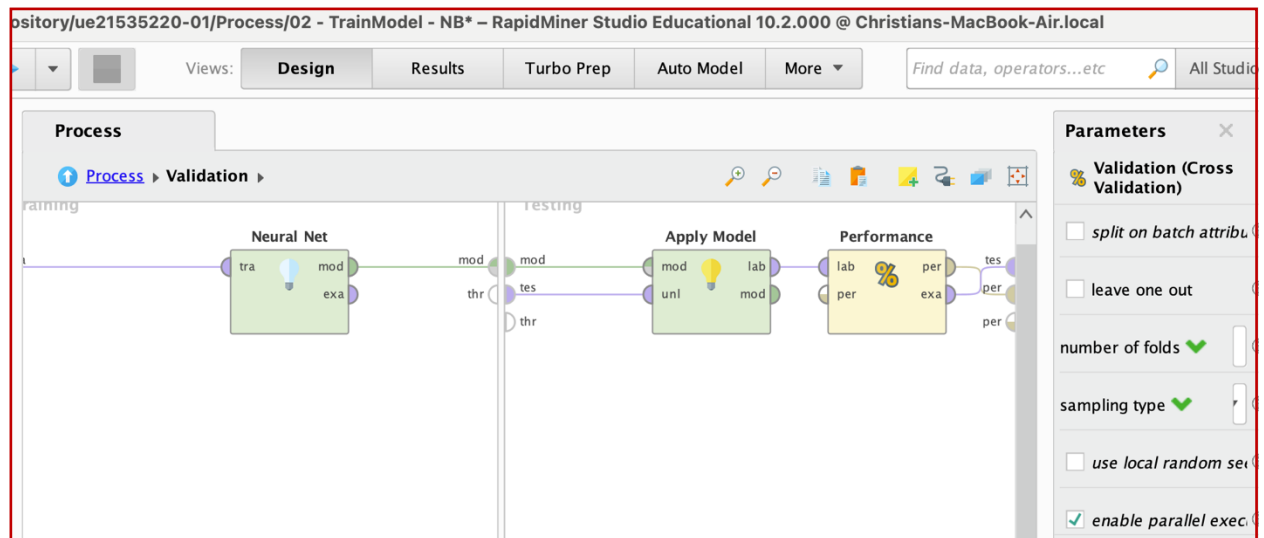
Turbo Prep
Filter (100 / 100 examples):

Row No.	topicId	word	weigh
1	0	iPhone	772
2	0	Apple	471
3	0	iPad	407
4	0	iOS	377
5	0	iphone	350
6	0	iCloud	264
7	0	phone	262
8	0	ios	260
9	0	back	240
10	0	get	226
11	1	screen	486
12	1	MacBook	418
13	1	Pro	381
14	1	problem	315
15	1	display	249

2.7 Modelado

Necesitamos construir y validar una clasificación multinivel para n-clases. La idea básica aquí es que cada clase debe tener un conjunto único de palabras clave asociadas con ese tipo de clase. Por ej. Los términos utilizados para los dispositivos Apple deberían ser diferentes para los dispositivos Android y los algoritmos de aprendizaje automático deberán poder diferenciarlos

Como se muestra a continuación, se utiliza el operador Cross-Validation para obtener nuestras conclusiones:



Finalmente, y tras realizar el modelado completo, obtenemos que el resultado de este análisis es de un 95% con el operador Cross-Validation, lo cual nos asegura que el modelado completo se ha realizado de manera correcta y que nuestro proyecto es concluyente.

	true 3DPrinting	true Android	true Apple	true DbA	true Unix	class precision
pred. 3DPrinting	1184	7	12	2	26	96.18%
pred. Android	12	1578	196	1	147	81.59%
pred. Apple	19	147	2076	13	242	83.14%
pred. DbA	9	6	29	2017	202	89.13%
pred. Unix	26	62	187	117	2683	87.25%
class recall	94.72%	87.67%	83.04%	93.81%	81.30%	