2025

Proyecto

José Miguel Grande Abrajan	202123224
Orquídea Rubí Palacios Godínez	202246635
Mancilla Carbajal Miguel Antonio	202242736

1-12-2025

Resumen

El presente documento detalla la ejecución de un proyecto integral de Minería de Datos utilizando la suite de software Weka. El objetivo principal fue aplicar y evaluar técnicas de preprocesamiento, clasificación supervisada y agrupamiento no supervisado sobre la base de datos "optdigits", la cual contiene información numérica sobre el reconocimiento óptico de dígitos manuscritos.

La metodología empleada abarcó tres fases principales: primero, la reducción de dimensionalidad mediante algoritmos de extracción y selección de características (PrincipalComponents, CorrelationAttributeEval y GainRatioAttributeEval); segundo, la evaluación comparativa de tres clasificadores (J48, NaiveBayes e IBk) sobre las bases de datos original y reducidas; y tercero, la aplicación de técnicas de clustering ignorando la etiqueta de clase.

Para validar los resultados, se utilizaron las herramientas Explorer, Experimenter y KnowledgeFlow. Los resultados estadísticos (T-Test) demostraron que el algoritmo IBk (K-Vecinos Cercanos) ofrece el mejor rendimiento para este dominio, alcanzando una precisión superior al 98%. Asimismo, se concluyó que las técnicas de selección de atributos (Correlation y GainRatio) son efectivas para reducir el costo computacional sin sacrificar la capacidad predictiva de los modelos, a diferencia del análisis de componentes principales (PCA), que en este caso específico presentó una leve disminución en la precisión.

Introducción

En el ámbito de la Ciencia de Datos y el Aprendizaje Automático (Machine Learning), contar con plataformas que permitan preprocesar información y evaluar modelos de manera eficiente es crucial. Para el desarrollo de este proyecto, se ha utilizado Weka (Waikato Environment for Knowledge Analysis), una suite de software de código abierto desarrollada en la Universidad de Waikato, Nueva Zelanda. Escrito en Java, Weka integra una colección completa de algoritmos de aprendizaje automático para tareas de minería de datos, permitiendo al usuario aplicar métodos de preprocesamiento, clasificación, regresión, agrupamiento (clustering) y reglas de asociación directamente sobre sus conjuntos de datos.

Para cumplir con los objetivos de este análisis, hemos hecho uso de las tres interfaces principales que ofrece este entorno:

- **Weka Explorer (Explorador):** Es la interfaz principal y más utilizada para el análisis exploratorio de datos. Funciona como un entorno de "laboratorio" donde se cargan los datos crudos para procesarlos paso a paso. En este proyecto, el Explorer fue fundamental para la fase de Preprocesamiento (limpieza y reducción de dimensiones),

así como para realizar pruebas iniciales rápidas de los algoritmos de clasificación y agrupamiento. Nos permite visualizar la distribución de los atributos y evaluar el rendimiento de los modelos de forma individual e inmediata mediante matrices de confusión y estadísticas de error.

- **Weka Experimenter (Experimentador):** A diferencia del Explorer, que evalúa un modelo a la vez, el Experimenter es una herramienta diseñada para la automatización y la validación estadística rigurosa. Su función principal en este proyecto fue permitirnos comparar objetivamente el desempeño de múltiples algoritmos (como J48, NaiveBayes e IBk) sobre diferentes versiones de la base de datos (original y reducidas) de manera simultánea. Esta herramienta ejecuta pruebas de validación cruzada y aplica tests de significancia estadística (T-tests) para determinar si las diferencias en la precisión de los algoritmos son producto del azar o si un modelo es matemáticamente superior a otro.
- **Weka KnowledgeFlow (Flujo de Conocimiento):** Esta interfaz ofrece una perspectiva visual del proceso de minería de datos basada en el paradigma de "flujo de datos". En lugar de pestañas estáticas, utiliza un lienzo donde el usuario arrastra y conecta componentes (como cargadores de datos, filtros y evaluadores) para diseñar la tubería de procesamiento. En nuestra práctica, el KnowledgeFlow se utilizó para modelar gráficamente y replicar el procedimiento completo de agrupamiento (Clustering), demostrando cómo los datos fluyen desde su carga, pasan por la eliminación de la clase (filtro) y llegan al entrenamiento y evaluación del algoritmo, ofreciendo una representación clara de la arquitectura del proceso.

Descripción de las Técnicas de Minería de Datos

Para el análisis de la información contenida en la base de datos, se han implementado tres familias fundamentales de técnicas de minería de datos, cada una con un propósito específico en el flujo de descubrimiento de conocimiento:

1. Extracción y Selección de Características (Feature Extraction/Selection) Esta técnica, también conocida como reducción de dimensionalidad, consiste en el proceso de transformar o seleccionar un subconjunto de los atributos más relevantes de la base de datos original. En conjuntos de datos con un gran número de variables (como es el caso de optdigits), mucha información puede ser redundante, irrelevante o constituir "ruido" que confunde a los modelos. El objetivo de esta técnica es reducir la complejidad del espacio de datos sin perder la información esencial. Al aplicar algoritmos como el Análisis de Componentes Principales (PCA) o evaluaciones de correlación, logramos generar bases de datos "reducidas" que permiten un entrenamiento más rápido y, a menudo, modelos más precisos al eliminar variables confusas.

2. Clasificación (Classification) La clasificación es una técnica de Aprendizaje Supervisado. Su objetivo es construir un modelo matemático capaz de predecir la etiqueta de clase de una instancia nueva basándose en sus atributos. Para que esto funcione, el algoritmo "aprende" primero analizando un conjunto de datos de entrenamiento donde ya se conoce la respuesta

correcta (en nuestro caso, saber qué dígito representa cada imagen). Una vez entrenado, el modelo busca patrones generales que relacionan los atributos numéricos con la clase objetivo. En este proyecto, se aplicaron algoritmos de clasificación sobre las bases de datos original y reducidas para verificar qué tan bien podía el sistema distinguir entre los diferentes dígitos manuscritos.

3. Agrupamiento (Clustering) A diferencia de la clasificación, el agrupamiento es una técnica de Aprendizaje No Supervisado. Esto significa que el algoritmo no cuenta con una etiqueta de clase predefinida ni un "maestro" que le indique la respuesta correcta. El objetivo del clustering es explorar la estructura natural de los datos para encontrar grupos (clústers) de instancias que sean muy similares entre sí, pero muy diferentes a las instancias de otros grupos. En nuestra práctica, para aplicar correctamente esta técnica, fue necesario ignorar o remover el atributo de la clase, forzando al algoritmo a agrupar los datos basándose puramente en la similitud matemática de sus atributos.

Descripción de los Algoritmos Utilizados

A continuación, se describen los fundamentos teóricos de los algoritmos seleccionados para cada etapa del proceso de minería de datos en Weka.

1. Algoritmos para la Extracción y Selección de Características

Para reducir la dimensionalidad de la base de datos optdigits y optimizar el rendimiento computacional, se utilizaron tres evaluadores distintos acompañados del método de búsqueda **Ranker**:

- **PrincipalComponents (Análisis de Componentes Principales - PCA):**

Este algoritmo no simplemente selecciona atributos, sino que transforma el espacio de datos original. Realiza una transformación lineal ortogonal para convertir un conjunto de observaciones de variables posiblemente correlacionadas en un conjunto de valores de variables linealmente no correlacionadas llamadas "componentes principales". Su objetivo es proyectar los datos en un espacio de menor dimensión maximizando la varianza de los datos, conservando así la mayor cantidad de información posible con el menor número de atributos nuevos.

- **CorrelationAttributeEval:**

Este algoritmo evalúa la importancia de un atributo midiendo la correlación (generalmente de Pearson) entre este y la clase objetivo. La premisa es que los buenos atributos están altamente correlacionados con la clase, pero no correlacionados entre sí. Junto con el método de búsqueda Ranker, ordena los atributos desde el más correlacionado al menos correlacionado, permitiendo descartar aquellos que tienen una relación débil con el resultado.

- **GainRatioAttributeEval:**

Basado en la teoría de la información, este algoritmo evalúa el valor de un atributo calculando su "Ratio de Ganancia" (Gain Ratio) con respecto a la clase. Es una mejora sobre la medida de Information Gain (Ganancia de Información) que compensa el sesgo hacia atributos que tienen muchos valores distintos. Mide cuánto disminuye la entropía (incertidumbre) del sistema al conocer el valor del atributo.

2. Algoritmos de Clasificación

Para la tarea de predicción supervisada de los dígitos, se eligieron tres algoritmos representativos de diferentes paradigmas de aprendizaje:

- **J48 (Árboles de Decisión C4.5):**

J48 es la implementación en Java del famoso algoritmo C4.5. Es un clasificador basado en árboles de decisión. Funciona dividiendo recursivamente el conjunto de datos basándose en el atributo que proporciona la mayor ganancia de información normalizada en cada nodo. El resultado es una estructura en forma de árbol similar a un diagrama de flujo, donde cada nodo

interno representa una prueba en un atributo, cada rama representa el resultado de la prueba y cada hoja representa una etiqueta de clase. Es altamente interpretable por humanos.

- **NaiveBayes (Bayes Ingenuo):**

Es un clasificador probabilístico fundamentado en el Teorema de Bayes. Se denomina "ingenuo" (Naive) porque asume una fuerte independencia entre los atributos; es decir, asume que la presencia de una característica particular en una clase no está relacionada con la presencia de ninguna otra característica. A pesar de esta simplificación teórica, es extremadamente rápido y muy efectivo en problemas de alta dimensión como el reconocimiento de patrones.

- **IBk:**

Conocido comúnmente como K-Vecinos Más Cercanos (KNN). Es un algoritmo de aprendizaje "vago" (lazy learning), lo que significa que no construye un modelo interno explícito durante el entrenamiento. Simplemente almacena las instancias de entrenamiento y, para clasificar una nueva instancia, busca las "k" instancias más cercanas (usando una métrica de distancia, como la Euclidiana) en el espacio de características y asigna la clase más común entre esos vecinos.

3. Algoritmos de Agrupamiento (Clustering)

Para el análisis no supervisado, se utilizaron algoritmos particionales para encontrar estructuras subyacentes en los datos sin usar la etiqueta de clase:

- **SimpleKMeans:**

Es uno de los algoritmos de clustering más utilizados. Su objetivo es particionar las n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio (centroide) es el más cercano. El algoritmo funciona iterativamente: primero asigna centroides aleatorios, luego asigna cada punto a su centroide más cercano y finalmente recalcula los centroides basándose en el promedio de los puntos asignados, repitiendo el proceso hasta que los centroides se estabilizan.

- **FarthestFirst:**

Es una variante heurística del algoritmo K-Means diseñada para la velocidad. En lugar de seleccionar centroides aleatorios iniciales y reajustarlos iterativamente, coloca el primer centroide aleatoriamente y selecciona cada centroide subsiguiente como el punto que está más alejado de los centroides ya existentes. Es extremadamente rápido, ideal para bases de datos grandes o como paso preliminar.

- **Canopy:**

Es un algoritmo de agrupamiento a menudo utilizado como un "preprocesador" para K-Means. Utiliza dos umbrales de distancia (T1 y T2) para crear grupos superpuestos llamados

"canopies". Es muy eficiente computacionalmente y ayuda a estimar el número óptimo de clústers o a proporcionar puntos de inicio más inteligentes para algoritmos más costosos como K-Means.

Descripción de la Base de Datos Utilizada

Para la realización de este proyecto experimental, se seleccionó la base de datos "Optical Recognition of Handwritten Digits" (optdigits), un conjunto de datos clásico y ampliamente utilizado en la comunidad de aprendizaje automático, disponible originalmente en el repositorio de la UCI (University of California, Irvine).

Características Técnicas:

- **Volumen de Datos:** La base de datos contiene un total de **5,**620 instancias (muestras). Este tamaño fue seleccionado estratégicamente para el proyecto, ya que proporciona suficientes datos para validar estadísticamente los resultados sin comprometer el tiempo de procesamiento limitado para la ejecución de múltiples algoritmos.
- **Dimensionalidad:** Cuenta con 64 atributos de entrada (predictivos) y 1 atributo de clase.
- **Tipo de Atributos:** Todos los atributos de entrada son **numéricos (enteros)**.

Atributo de Clase (Target):

El objetivo de la base de datos es identificar el dígito escrito. Por lo tanto, el atributo de clase es nominal y posee 10 valores posibles: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Justificación de la Elección:

Se eligió esta base de datos por dos razones principales:

1. **Naturaleza Numérica:** Al ser una base de datos puramente numérica, facilita la aplicación directa de algoritmos matemáticos como *PrincipalComponents* y *K-Means* sin necesidad de transformaciones complejas previas.
2. **Alta Dimensionalidad:** Al tener 64 atributos, es un candidato ideal para demostrar la efectividad de las técnicas de **reducción de características** (paso clave de este proyecto), permitiendo observar claramente cómo se puede comprimir la información manteniendo la capacidad predictiva.

Descripción de la Práctica y Procedimiento Experimental

El desarrollo del proyecto se llevó a cabo siguiendo una metodología estructurada en cinco fases secuenciales, utilizando las diferentes herramientas proporcionadas por el entorno Weka. A continuación, se detalla el procedimiento técnico ejecutado en cada etapa:

Fase 1: Preprocesamiento y Reducción de Dimensionalidad (Weka Explorer)

Como paso inicial, se cargó la base de datos original optdigits.arff en el módulo **Explorer**. Con el objetivo de optimizar el conjunto de datos y eliminar redundancia entre los 64 atributos de entrada, se aplicaron tres técnicas distintas de selección y extracción de características:

1. **Reducción por PCA:** Se utilizó el evaluador PrincipalComponents junto con el método de búsqueda Ranker. Esto transformó el espacio de atributos original en nuevos componentes ortogonales. Se guardó el conjunto resultante como una nueva base de datos reducida.
2. **Reducción por Correlación:** Se aplicó el evaluador CorrelationAttributeEval para identificar los atributos con mayor correlación lineal respecto a la clase. Se generó un segundo archivo reducido conservando únicamente los atributos mejor rankeados.
3. **Reducción por Ratio de Ganancia:** Se empleó GainRatioAttributeEval para medir la ganancia de información de cada píxel. Se generó un tercer archivo reducido con esta selección.

Como resultado de esta fase, se obtuvieron cuatro versiones del conjunto de datos: la Original y tres versiones Reducidas (PCA, Correlation, GainRatio), listas para ser comparadas.

Fase 2: Ejecución de Modelos de Clasificación (Weka Explorer)

En esta etapa se evaluó el rendimiento predictivo. Para garantizar la validez de los resultados, se utilizó el esquema de evaluación Cross-Validation (Validación Cruzada) de 10 pliegues (folds), que asegura que el modelo sea probado en datos que no vio durante el entrenamiento.

Se ejecutaron tres algoritmos de clasificación de familias distintas: J48 (Árboles), NaiveBayes (Probabilístico) y IBk (Instancias). Este procedimiento se repitió sistemáticamente para cada una de las 4 bases de datos generadas en la fase anterior, registrando métricas de precisión (Accuracy) y matrices de confusión para su posterior análisis.

Fase 3: Validación Estadística Automatizada (Weka Experimenter)

Para dar rigor científico a la comparación, se trasladó el flujo de trabajo al entorno **Experimenter**.

- **Configuración:** Se definió un experimento cargando las 4 bases de datos (datasets) y los 3 algoritmos seleccionados. Se configuró la salida de resultados a un archivo .arff para su registro permanente.

- **Ejecución:** El sistema ejecutó automáticamente 10 repeticiones de la validación cruzada de 10 pliegues para cada combinación de algoritmo-dataset, generando una muestra estadística robusta.
- **Análisis:** Se aplicó una prueba de significancia estadística (Paired T-Test con un nivel de confianza del 95% o 0.05) utilizando como base de comparación el dataset original o el algoritmo estándar, permitiendo identificar matemáticamente qué combinaciones ofrecían una mejora real ("v") o un deterioro significativo ("*") en el rendimiento.

Fase 4: Análisis de Agrupamiento (Clustering)

Se procedió a realizar un análisis no supervisado utilizando la pestaña Clúster del Explorer.

- **Preparación:** Dado que el clustering busca patrones naturales sin etiquetas predefinidas, se configuró el entorno para ignorar el atributo de clase ("Ignore attributes") durante el entrenamiento.
- **Algoritmos:** Se aplicaron los algoritmos **SimpleKMeans**, **FarthestFirst** y **Canopy** sobre la base de datos original y las reducidas.
- **Evaluación:** Se utilizó el modo "Classes to clústers evaluation" para contrastar cómo los grupos formados matemáticamente coincidían con los dígitos reales de la base de datos.

Fase 5: Modelado de Flujo de Datos (KnowledgeFlow)

Finalmente, se diseñó un pipeline visual para replicar y automatizar el proceso de agrupamiento. Se construyó un diagrama de flujo conectando los siguientes componentes:

1. **ArffLoader:** Para la ingesta de datos.
2. **ClassAssigner / Remove Filter:** Configurado específicamente para eliminar el atributo de la clase (índice last), simulando un escenario real de clustering no supervisado.
3. **TrainingSetMaker:** Para generar los lotes de datos de entrenamiento.
4. **Clusterer (SimpleKMeans):** Para ejecutar el agrupamiento.
5. **ClustererPerformanceEvaluator:** Para medir la calidad de los clústers.
6. **TextViewer:** Para visualizar los resultados finales y los centroides generados.

Este flujo permitió verificar la reproducibilidad del experimento de manera gráfica y modular.

Algoritmo	Base Original (%)	Base Reducida PCA (%)	Base Reducida Corr (%)	Base Reducida Gain (%)
J48	*90.694 %*	87.3665 %	90.694 %	*90.694 %*
NaiveBayes	*91.3345 %*	*87.4911 %*	91.3345 %	*91.3345 %*
IBk (KNN)	98.6121 %	*96.0498 %*	*98.6121 %*	*98.6121 %*

```
Dataset                    (1) trees.J4 | (2) lazy. (3) bayes
-----------------------------------------------------------------
optdigits                (100)    90.52 |   98.70 v   91.39
'optdigits-weka.filters.u(100)    90.52 |   98.70 v   91.39
'optdigits-weka.filters.u(100)    90.52 |   98.70 v   91.39
'optdigits-weka.filters.s(100)    87.41 |   96.12 v   87.42
-----------------------------------------------------------------
                               (v/ /*) |   (4/0/0)   (0/4/0)
```

Comparación Estadística de los Modelos de Clasificación

Para validar rigurosamente el desempeño de los modelos, se ejecutó una prueba de significancia estadística (Paired T-Test) con un nivel de confianza del 95% utilizando el módulo Experimenter de Weka.

La configuración del experimento estableció al algoritmo **J48 (Árboles de Decisión)** como la **base de comparación** (Columna 1), contrastando su rendimiento contra **IBk** (Columna 2) y **NaiveBayes** (Columna 3).

Interpretación de los Resultados:

La tabla de resultados arrojó los siguientes hallazgos estadísticos (donde v indica una mejora significativa y el espacio vacío indica similitud estadística):

1. Análisis Comparativo de Algoritmos:

- J48 (Base) vs. IBk (K-Vecinos Cercanos):

El algoritmo IBk demostró una superioridad estadística contundente en todos los escenarios (marcado con el símbolo v). Mientras que J48 obtuvo una precisión promedio de ~90.5% en la base original, IBk alcanzó un 98.70%.

- *Interpretación:* Esto confirma que, para la base de datos optdigits (imágenes pixeladas), un enfoque basado en instancias y distancia (como IBk) es mucho más efectivo que un enfoque basado en reglas de decisión (árboles), logrando clasificar los dígitos casi a la perfección.

- J48 (Base) vs. NaiveBayes:

Al comparar J48 contra NaiveBayes, la ausencia de símbolos (v o *) indica que no existe una diferencia estadísticamente significativa entre ellos. Aunque NaiveBayes muestra números absolutos ligeramente superiores (91.39% frente a 90.52%), matemáticamente se considera un "empate técnico". Ambos algoritmos tienen un desempeño competente, pero inferior a IBk.

2. Impacto de las Técnicas de Reducción:

- Efectividad de Correlation y GainRatio:

Al observar las filas correspondientes a CorrelationAttributeEval y GainRatioAttributeEval, los resultados numéricos son idénticos a los de la base Original para los tres algoritmos.

- *Conclusión:* Esto indica una selección de características ideal. Se logró reducir la dimensionalidad de la base de datos eliminando atributos irrelevantes sin perder ni una fracción de capacidad predictiva.

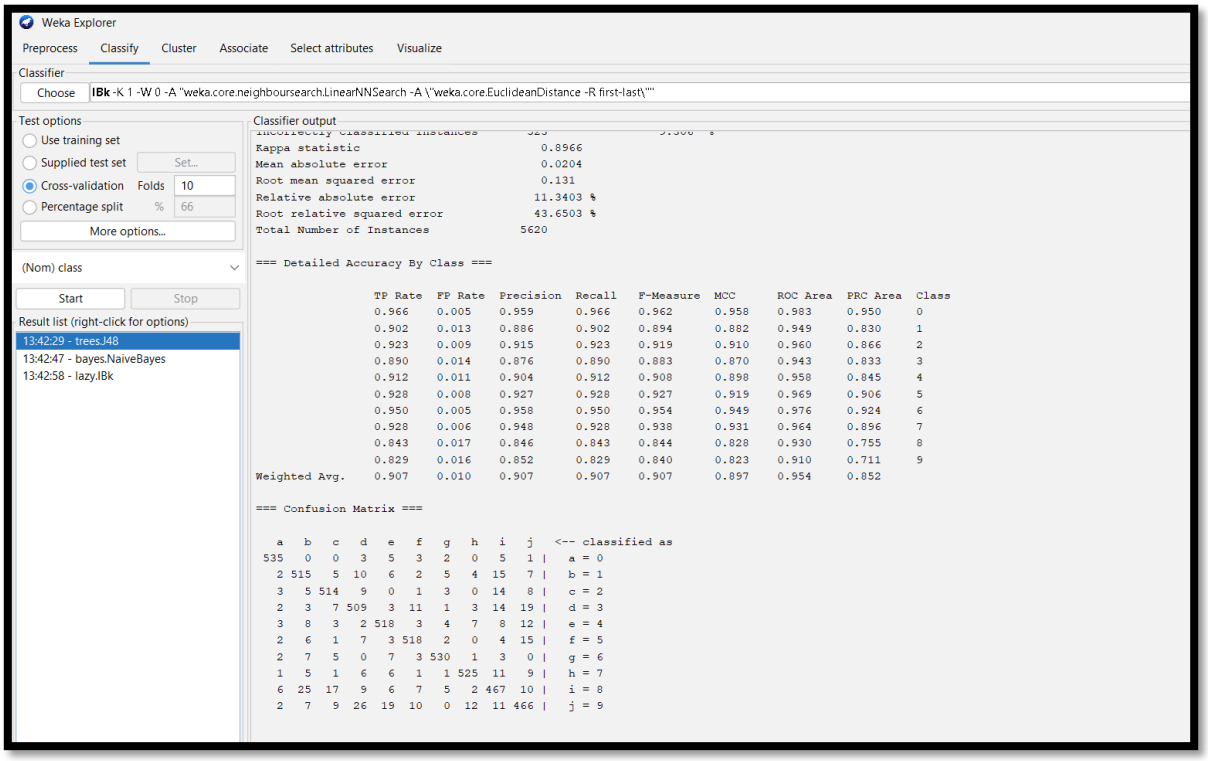
- Desempeño con PrincipalComponents (PCA):

La técnica PCA (última fila) provocó una caída generalizada en el rendimiento.

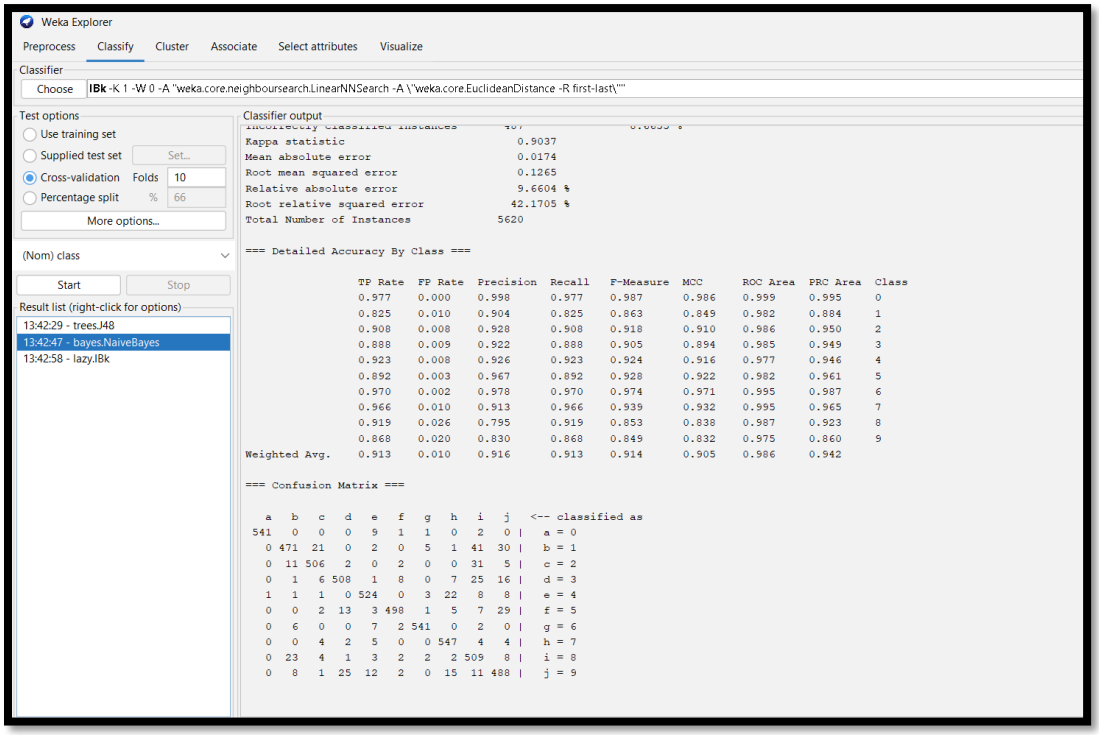
- El algoritmo base **J48** cayó de 90.52% a **87.41%**.
- El algoritmo **IBk** cayó de 98.70% a **96.12%**.
- *Conclusión:* La transformación geométrica de los datos realizada por PCA, en este caso específico, dificultó ligeramente la tarea de clasificación en comparación con trabajar con los píxeles originales o filtrados.

Optdigits.arff

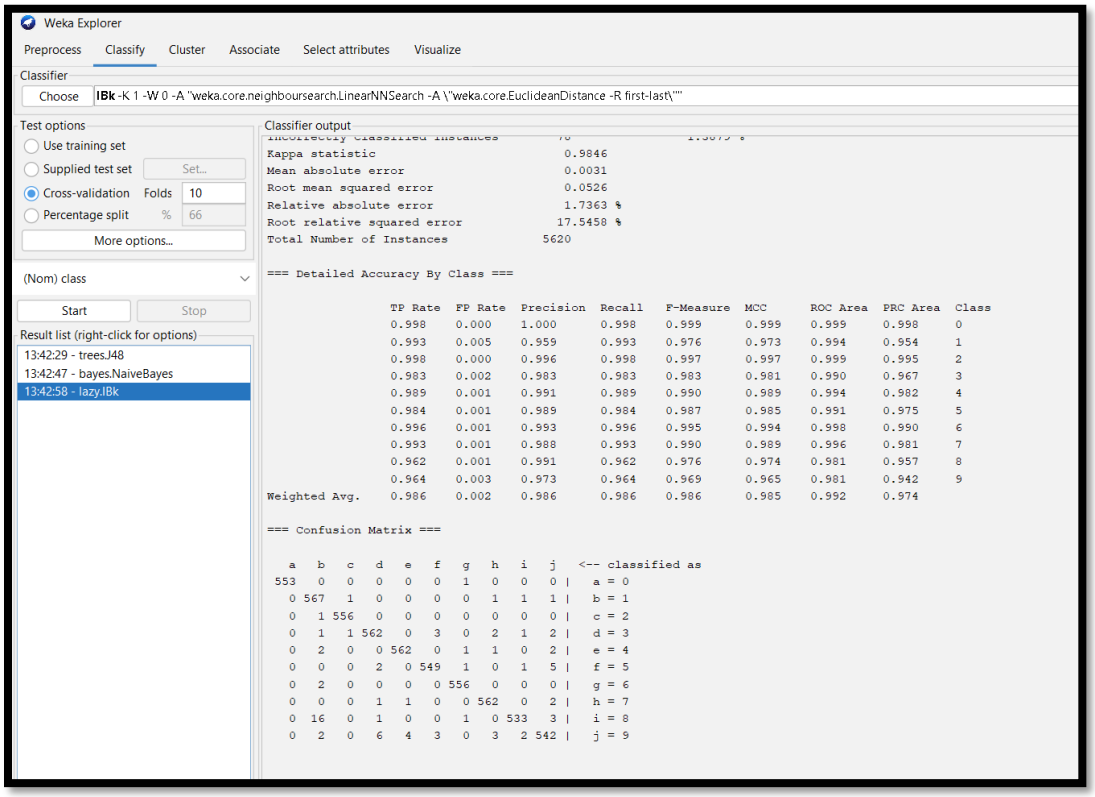
TREESJ48



NAIVEVAYES

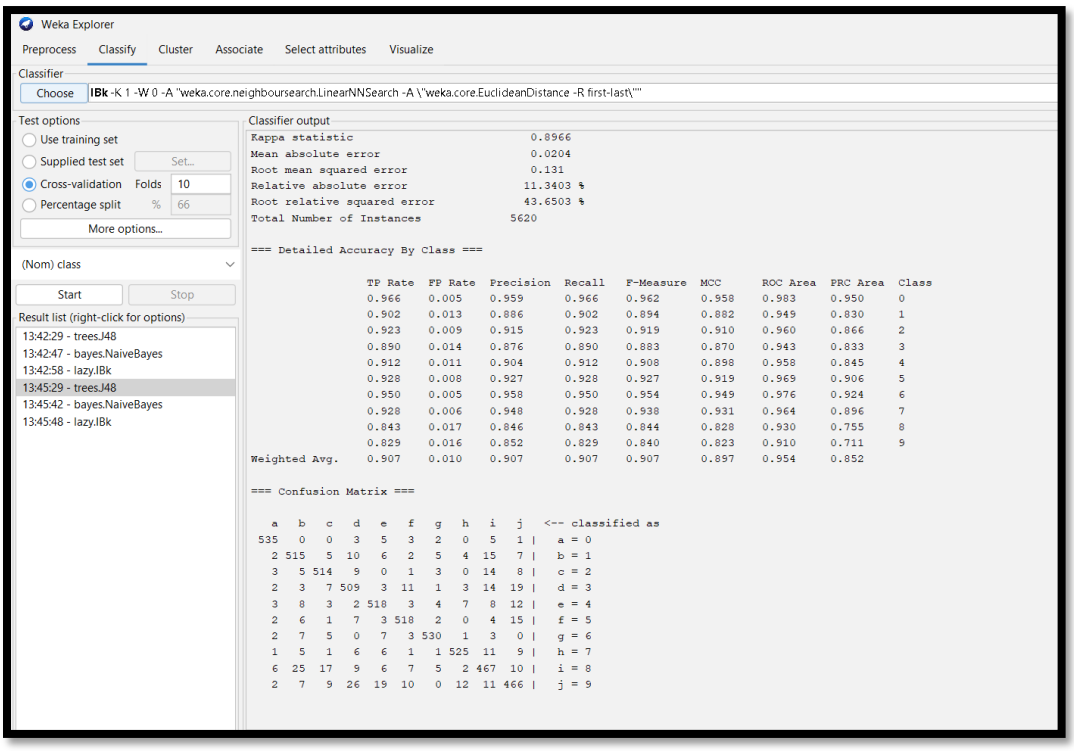


IBk

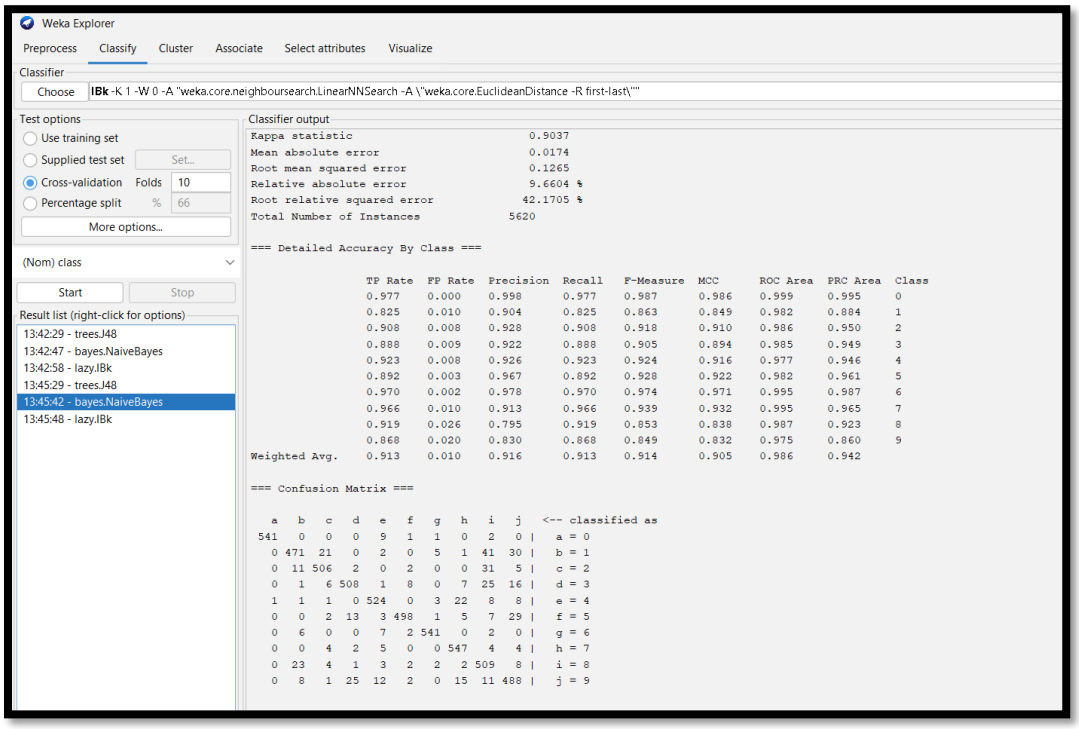


optdigits_CorrelationAttributeEval.arff

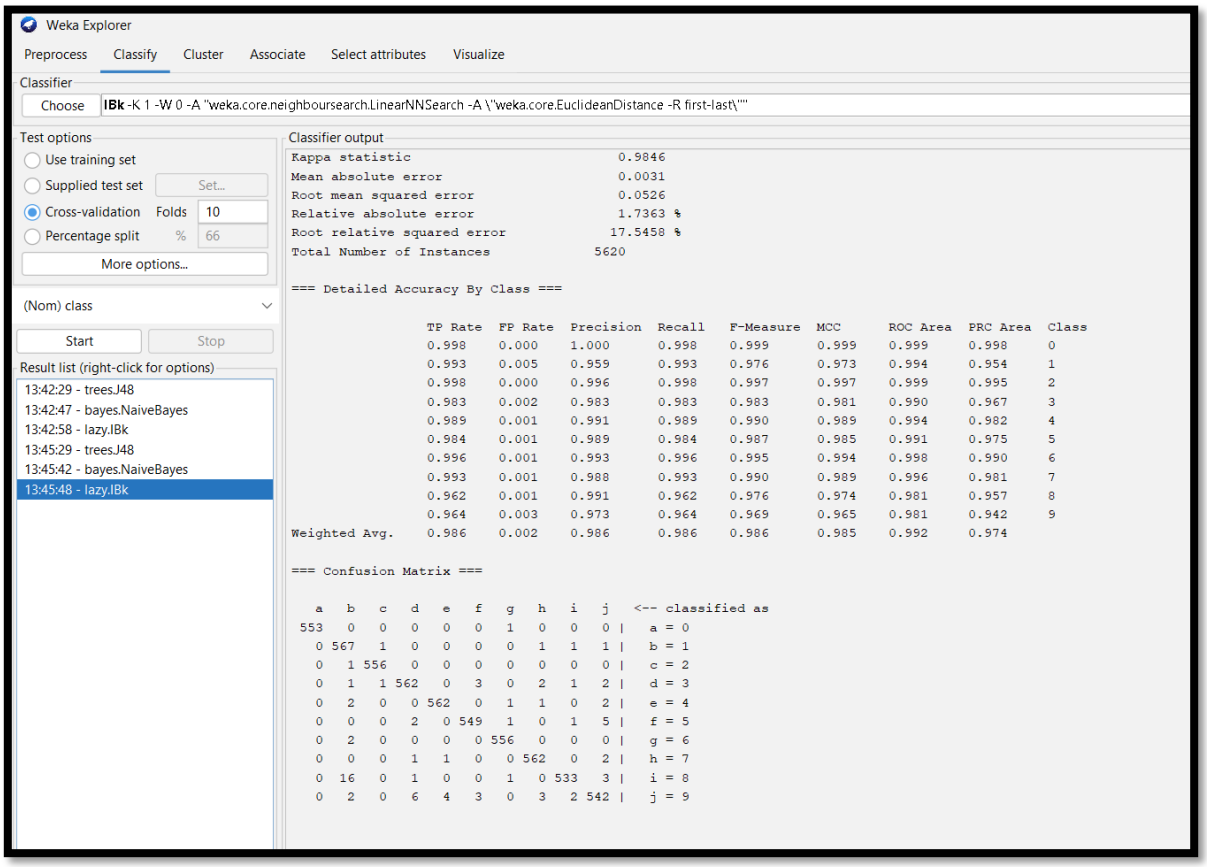
treesJ48



NaiveBayes

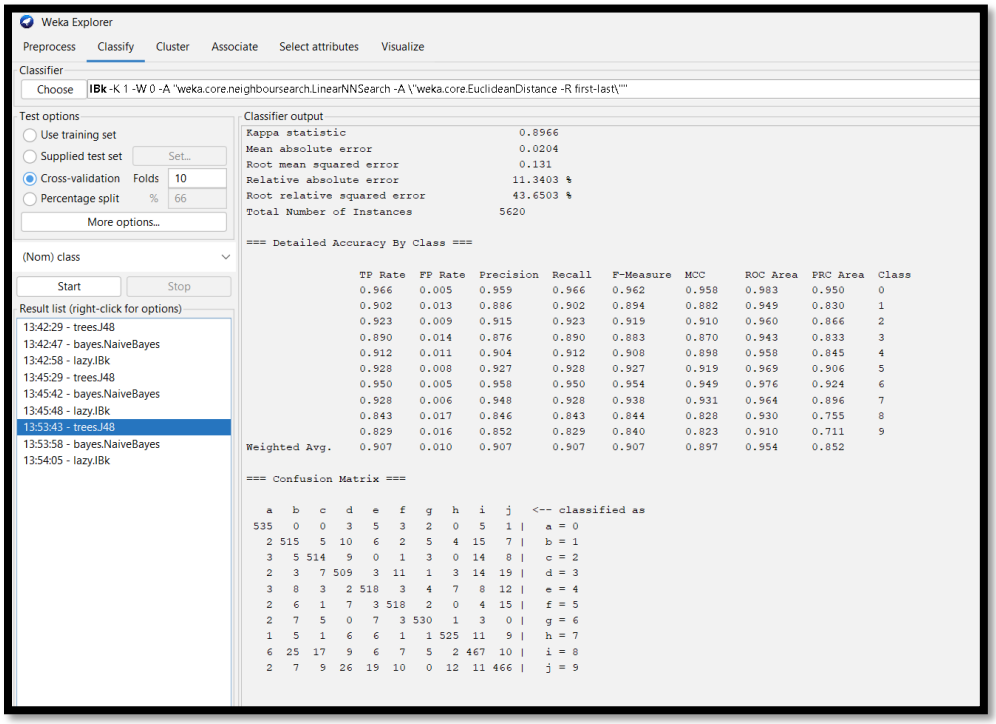


IBk

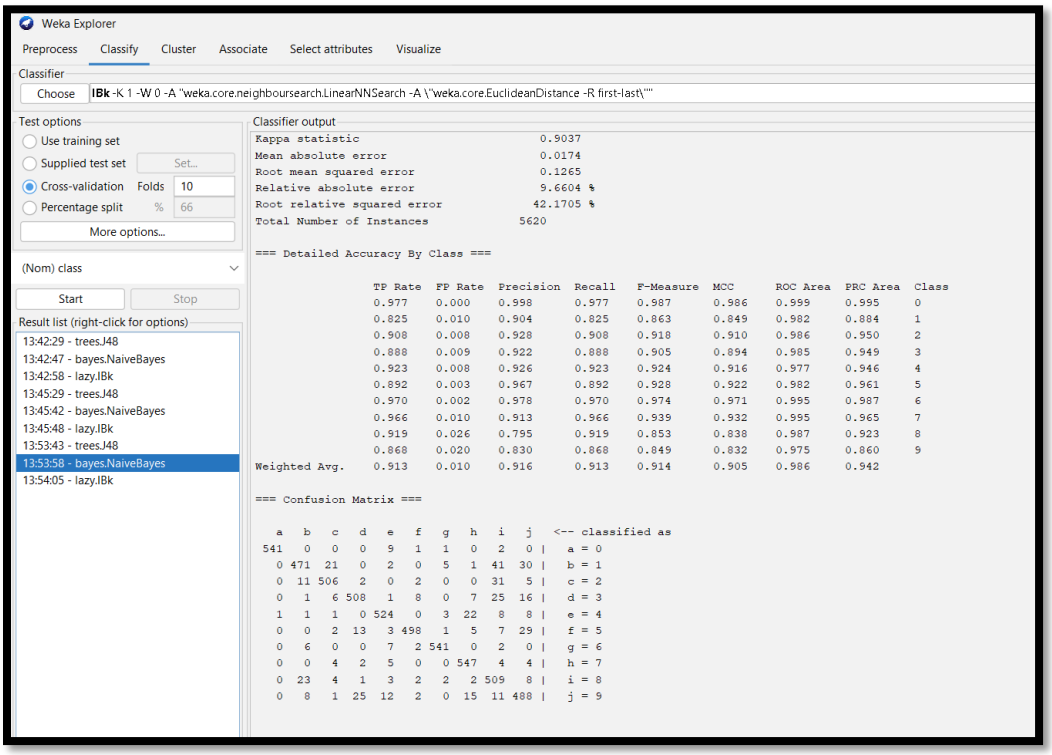

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
13:42:29 - trees.J48
13:42:47 - bayes.NaiveBayes
13:42:58 - lazy.IBk
13:45:29 - trees.J48
13:45:42 - bayes.NaiveBayes
13:45:48 - lazy.IBk
Classifier output
Kappa statistic 0.9846
Mean absolute error 0.0031
Root mean squared error 0.0526
Relative absolute error 1.7363 %
Root relative squared error 17.5458 %
Total Number of Instances 5620
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.998 0.000 1.000 0.998 0.999 0.999 0.999 0.998 0
0.993 0.005 0.959 0.993 0.976 0.973 0.994 0.954 1
0.998 0.000 0.996 0.998 0.997 0.997 0.999 0.995 2
0.983 0.002 0.983 0.983 0.983 0.981 0.990 0.967 3
0.989 0.001 0.991 0.989 0.990 0.989 0.994 0.982 4
0.984 0.001 0.989 0.984 0.987 0.985 0.991 0.975 5
0.996 0.001 0.993 0.996 0.995 0.994 0.998 0.990 6
0.993 0.001 0.988 0.993 0.990 0.989 0.996 0.981 7
0.962 0.001 0.991 0.962 0.976 0.974 0.981 0.957 8
0.964 0.003 0.973 0.964 0.969 0.965 0.981 0.942 9
Weighted Avg. 0.986 0.002 0.986 0.986 0.986 0.985 0.992 0.974
=== Confusion Matrix ===
a b c d e f g h i j <-- classified as
553 0 0 0 0 0 1 0 0 0 | a = 0
0 567 1 0 0 0 0 1 1 1 | b = 1
0 1 556 0 0 0 0 0 0 0 | c = 2
0 1 1 562 0 3 0 2 1 2 | d = 3
0 2 0 0 562 0 1 1 0 2 | e = 4
0 0 0 2 0 549 1 0 1 5 | f = 5
0 2 0 0 0 0 556 0 0 0 | g = 6
0 0 0 1 1 0 0 562 0 2 | h = 7
0 16 0 1 0 0 1 0 533 3 | i = 8
0 2 0 6 4 3 0 3 2 542 | j = 9

optdigits_GainRatioAttributeEval.arff

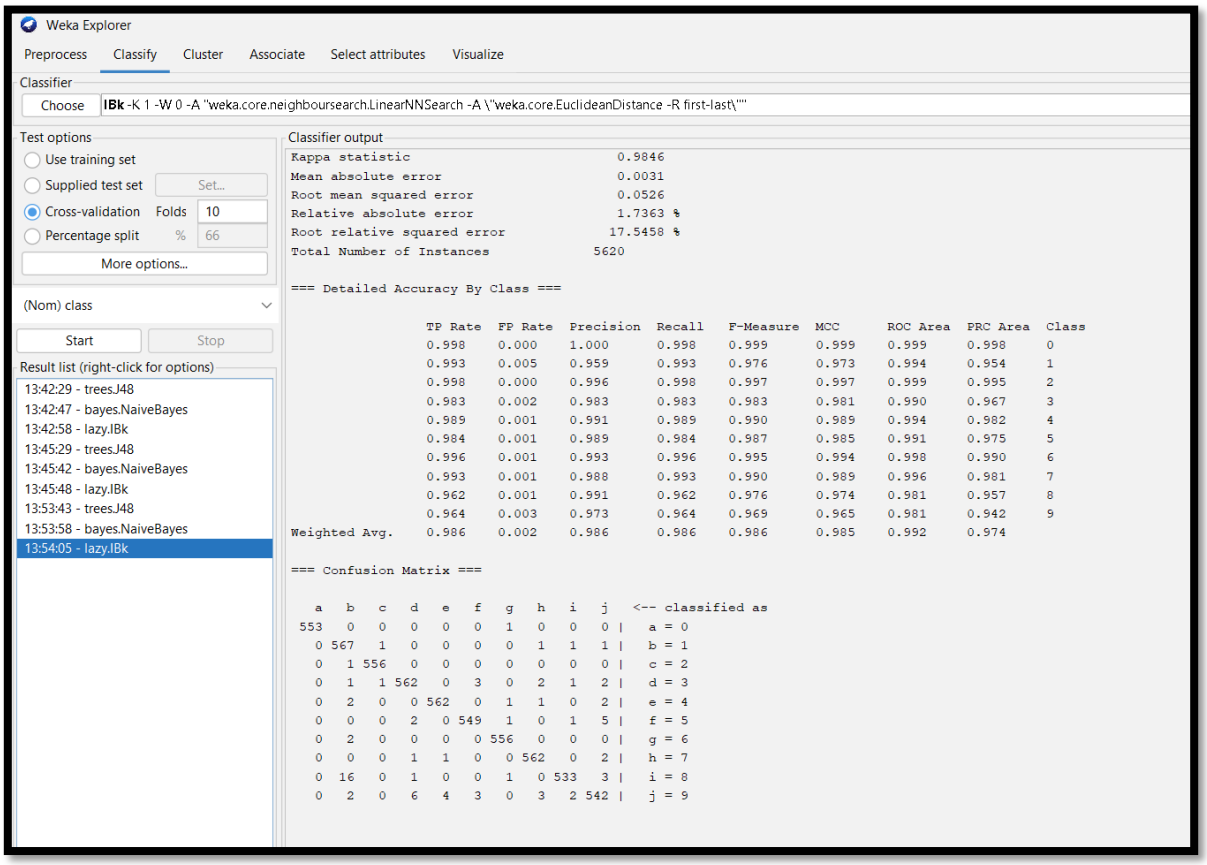
treesJ48



NaiveBayes

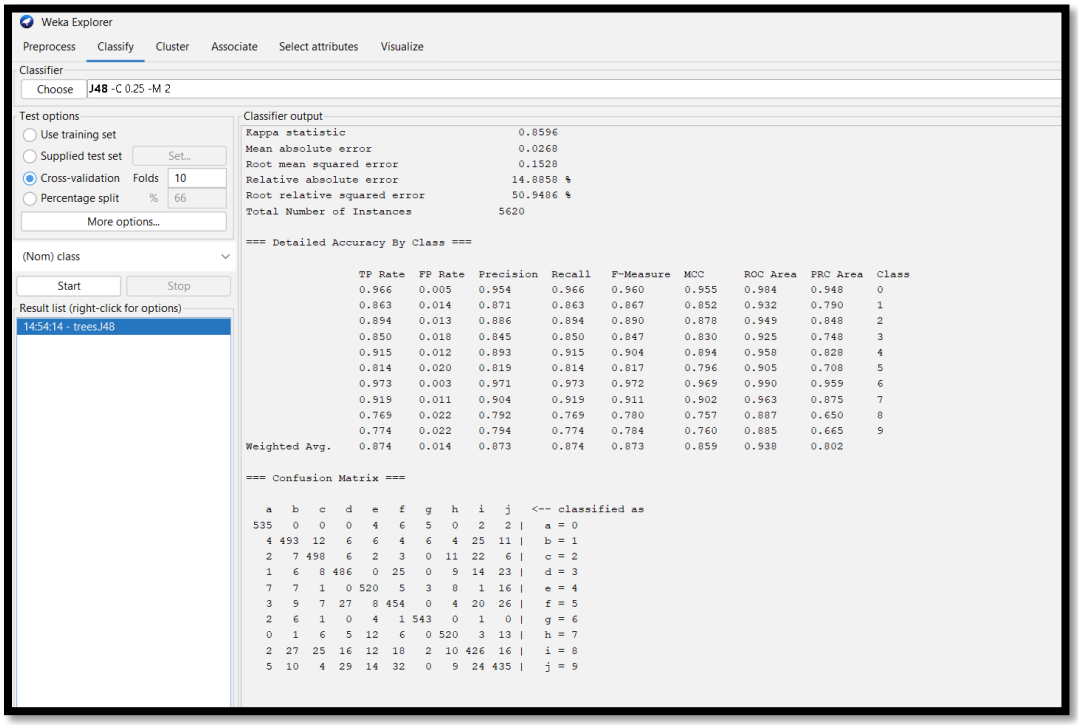


IBk

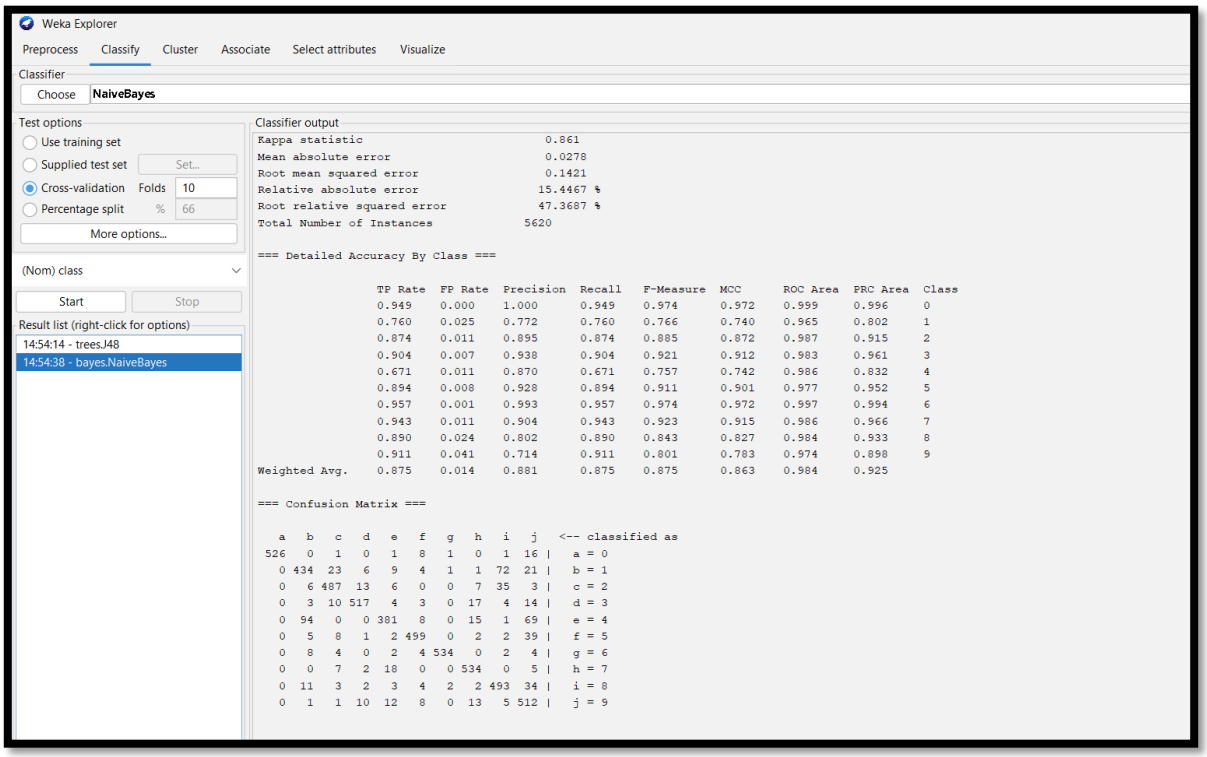

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
13:42:29 - trees.J48
13:42:47 - bayes.NaiveBayes
13:42:58 - lazy.IBk
13:45:29 - trees.J48
13:45:42 - bayes.NaiveBayes
13:45:48 - lazy.IBk
13:53:43 - trees.J48
13:53:58 - bayes.NaiveBayes
13:54:05 - lazy.IBk
Classifier output
Kappa statistic 0.9846
Mean absolute error 0.0031
Root mean squared error 0.0526
Relative absolute error 1.7363 %
Root relative squared error 17.5458 %
Total Number of Instances 5620
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.998 0.000 1.000 0.998 0.999 0.999 0.999 0.998 0
0.993 0.005 0.959 0.993 0.976 0.973 0.994 0.954 1
0.998 0.000 0.996 0.998 0.997 0.997 0.999 0.995 2
0.983 0.002 0.983 0.983 0.983 0.981 0.990 0.967 3
0.989 0.001 0.991 0.989 0.990 0.989 0.994 0.982 4
0.984 0.001 0.989 0.984 0.987 0.985 0.991 0.975 5
0.996 0.001 0.993 0.996 0.995 0.994 0.998 0.990 6
0.993 0.001 0.988 0.993 0.990 0.989 0.996 0.981 7
0.962 0.001 0.991 0.962 0.976 0.974 0.981 0.957 8
0.964 0.003 0.973 0.964 0.969 0.965 0.981 0.942 9
Weighted Avg. 0.986 0.002 0.986 0.986 0.986 0.985 0.992 0.974
=== Confusion Matrix ===
a b c d e f g h i j <-- classified as
553 0 0 0 0 0 1 0 0 0 | a = 0
0 567 1 0 0 0 0 1 1 1 | b = 1
0 1 556 0 0 0 0 0 0 0 | c = 2
0 1 1 562 0 3 0 2 1 2 | d = 3
0 2 0 0 562 0 1 1 0 2 | e = 4
0 0 0 2 0 549 1 0 1 5 | f = 5
0 2 0 0 0 0 556 0 0 0 | g = 6
0 0 0 1 1 0 0 562 0 2 | h = 7
0 16 0 1 0 0 1 0 533 3 | i = 8
0 2 0 6 4 3 0 3 2 542 | j = 9

optdigits_Principal Components.arff

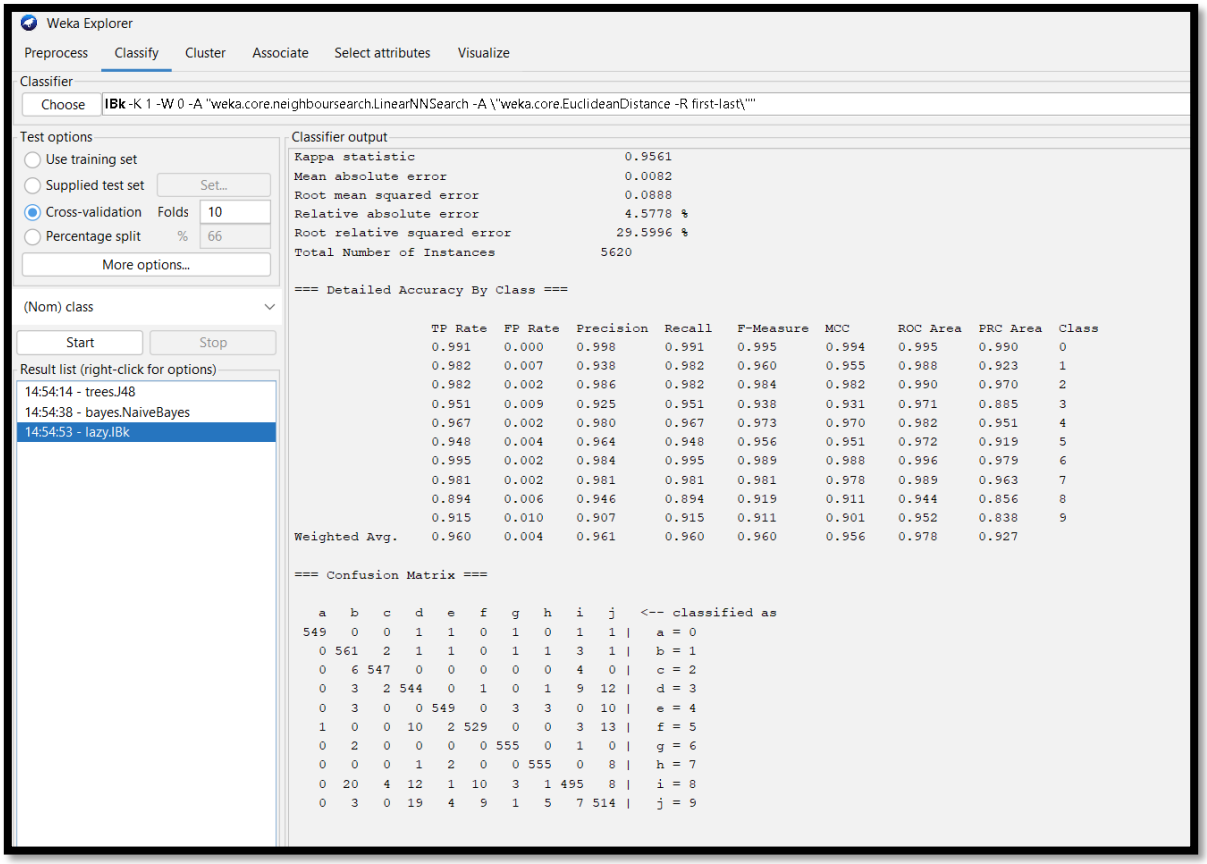
treesJ48



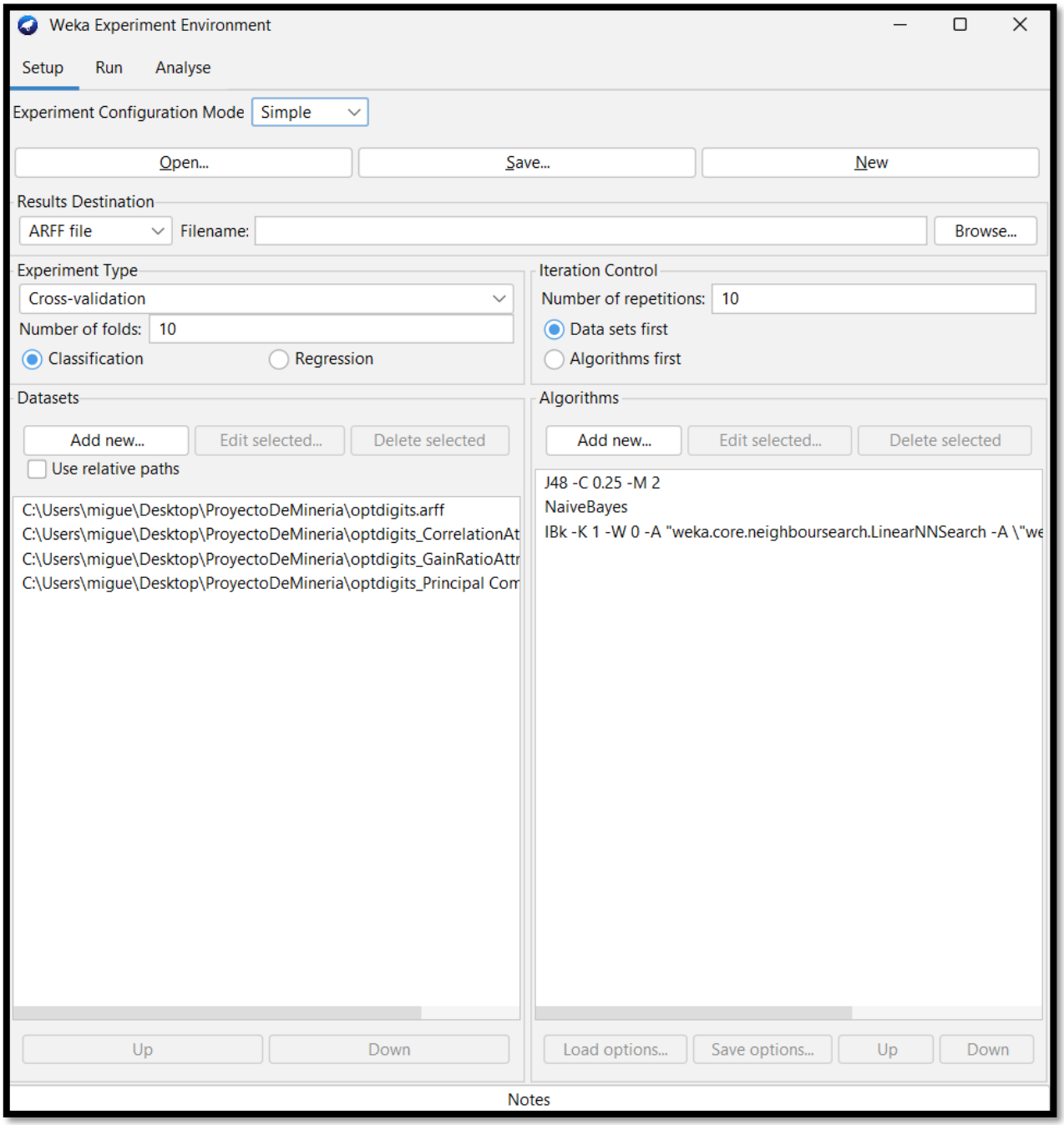
Naivevayes

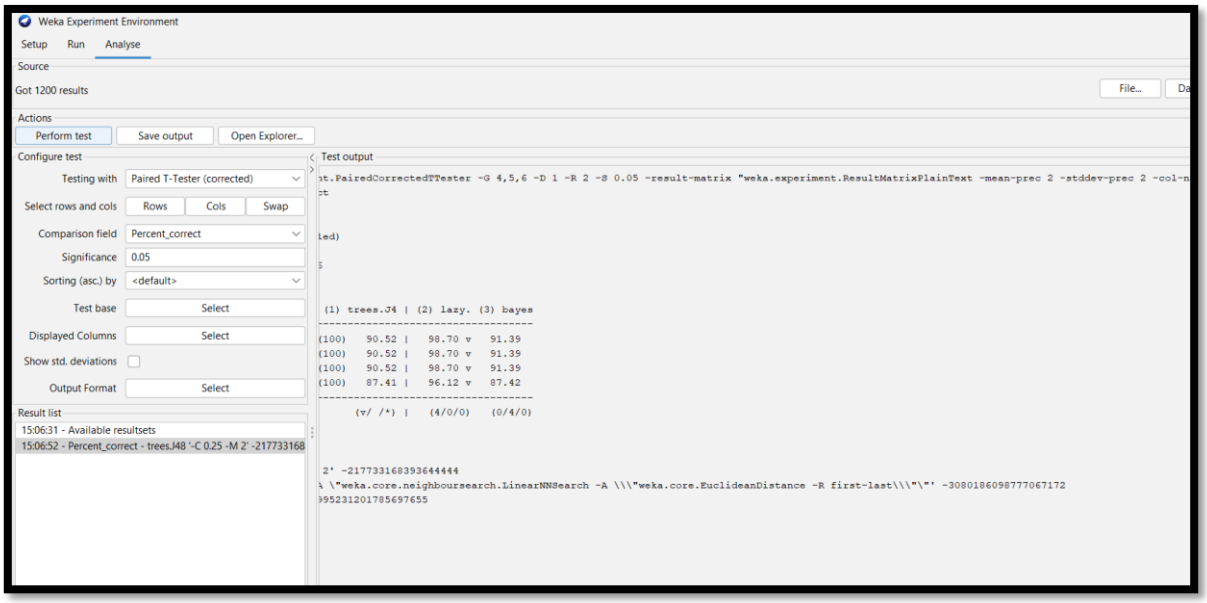


IBk


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
14:54:14 - trees.J48
14:54:38 - bayes.NaiveBayes
14:54:53 - lazy.IBk
Classifier output
Kappa statistic 0.9561
Mean absolute error 0.0082
Root mean squared error 0.0888
Relative absolute error 4.5778 %
Root relative squared error 29.5996 %
Total Number of Instances 5620
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
0.991 0.000 0.998 0.991 0.995 0.994 0.995 0.990 0
0.982 0.007 0.938 0.982 0.960 0.955 0.988 0.923 1
0.982 0.002 0.986 0.982 0.984 0.982 0.990 0.970 2
0.951 0.009 0.925 0.951 0.938 0.931 0.971 0.885 3
0.967 0.002 0.980 0.967 0.973 0.970 0.982 0.951 4
0.948 0.004 0.964 0.948 0.956 0.951 0.972 0.919 5
0.995 0.002 0.984 0.995 0.989 0.988 0.996 0.979 6
0.981 0.002 0.981 0.981 0.981 0.978 0.989 0.963 7
0.894 0.006 0.946 0.894 0.919 0.911 0.944 0.856 8
0.915 0.010 0.907 0.915 0.911 0.901 0.952 0.838 9
Weighted Avg. 0.960 0.004 0.961 0.960 0.960 0.956 0.978 0.927
=== Confusion Matrix ===
a b c d e f g h i j <-- classified as
549 0 0 1 1 0 1 0 1 1 | a = 0
0 561 2 1 1 0 1 1 3 1 | b = 1
0 6 547 0 0 0 0 0 4 0 | c = 2
0 3 2 544 0 1 0 1 9 12 | d = 3
0 3 0 0 549 0 3 3 0 10 | e = 4
1 0 0 10 2 529 0 0 3 13 | f = 5
0 2 0 0 0 0 555 0 1 0 | g = 6
0 0 0 1 2 0 0 555 0 8 | h = 7
0 20 4 12 1 10 3 1 495 8 | i = 8
0 3 0 19 4 9 1 5 7 514 | j = 9

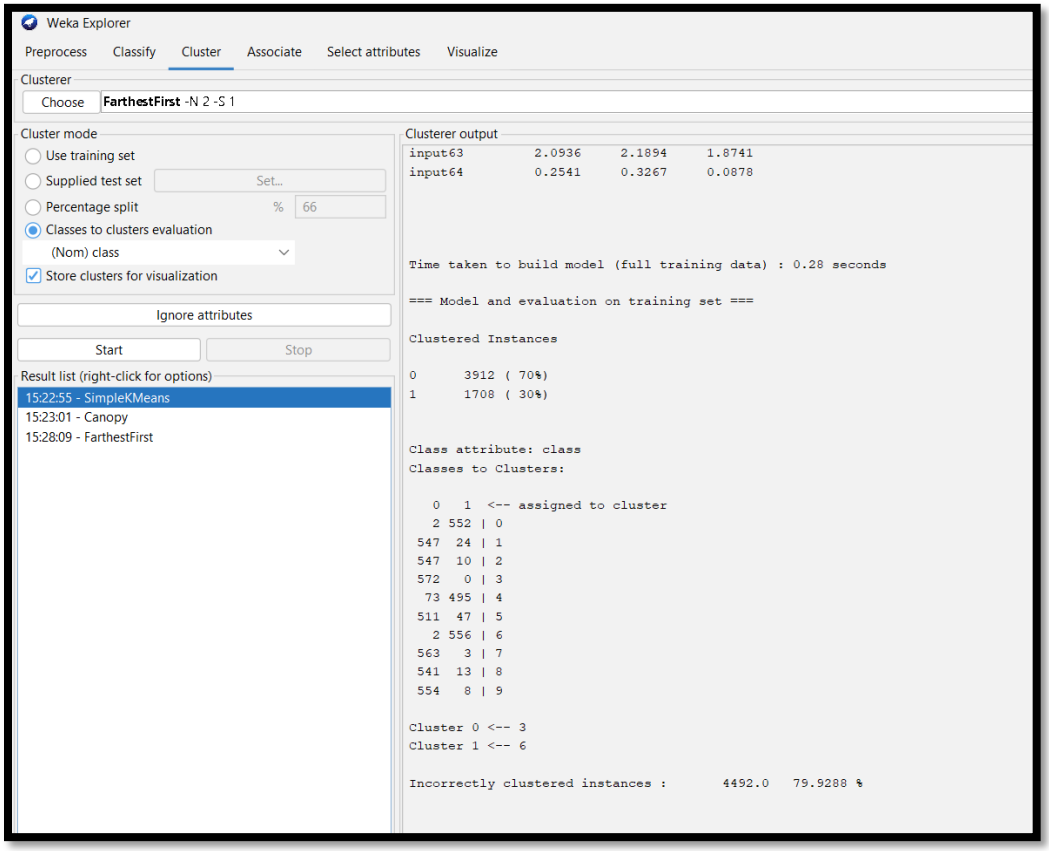
EXPERIMENTAL

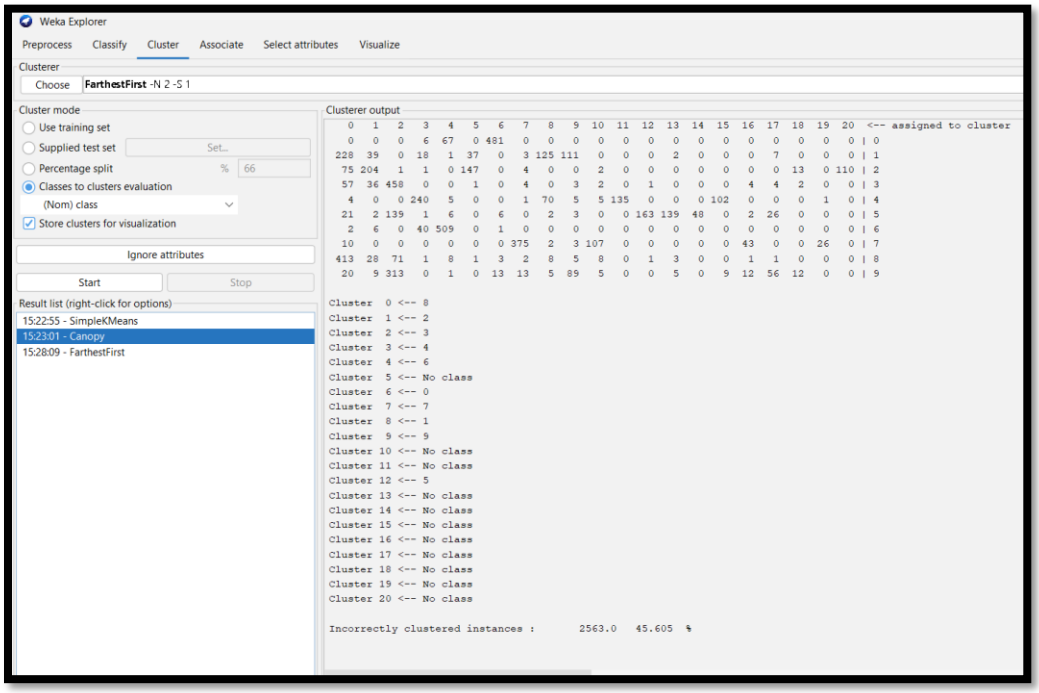



Weka Experiment Environment
Setup
Run
Analyse
Source
Got 1200 results
File...
Actions
Perform test
Save output
Open Explorer...
Configure test
Testing with
Paired T-Tester (corrected)
Select rows and cols
Rows
Cols
Swap
Comparison field
Percent_correct
Significance
0.05
Sorting (asc.) by
<default>
Test base
Select
Displayed Columns
Select
Show std. deviations
Output Format
Select
Result list
15:06:31 - Available resultsets
15:06:52 - Percent_correct - trees.J48 '-C 0.25 -M 2' -217733168
Test output
(1) trees.J4 | (2) lazy. (3) bayes
(100) 90.52 | 98.70 v 91.39
(100) 90.52 | 98.70 v 91.39
(100) 90.52 | 98.70 v 91.39
(100) 87.41 | 96.12 v 87.42
(v/ /*) | (4/0/0) (0/4/0)

Clustering

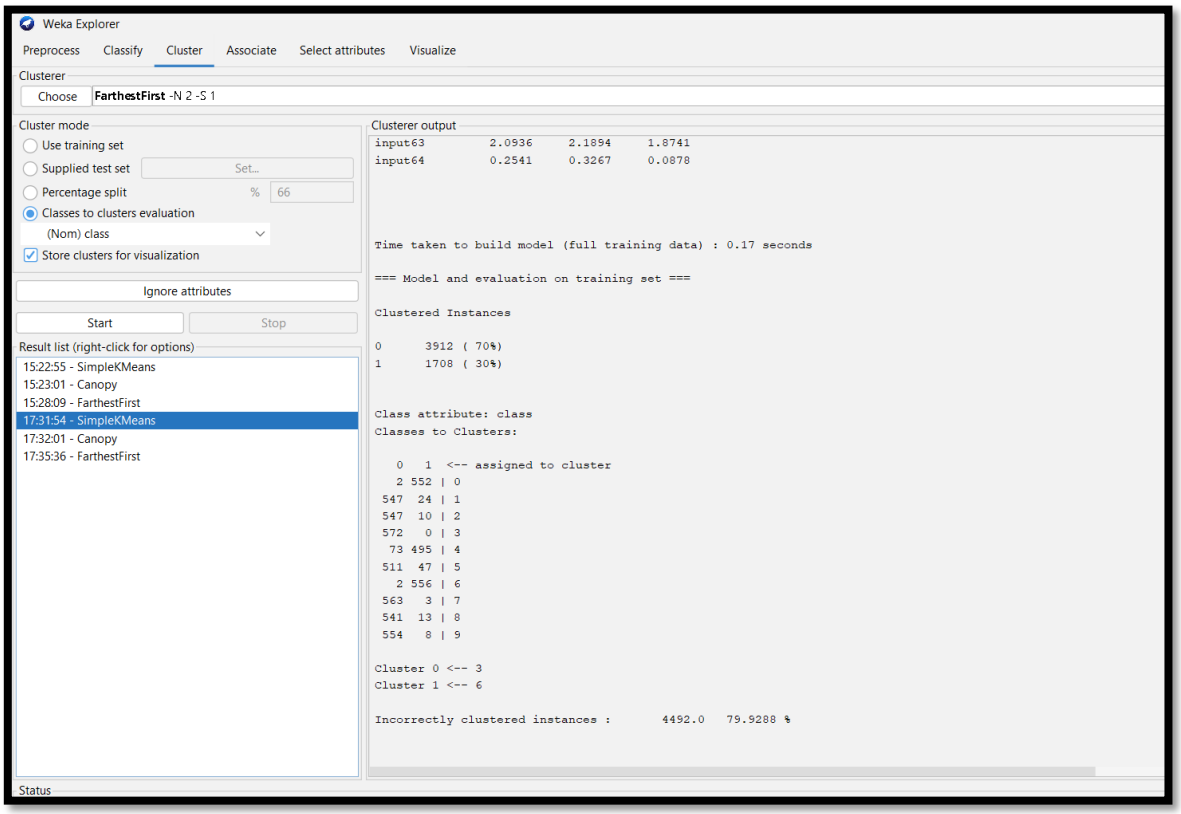
optdigits.arff

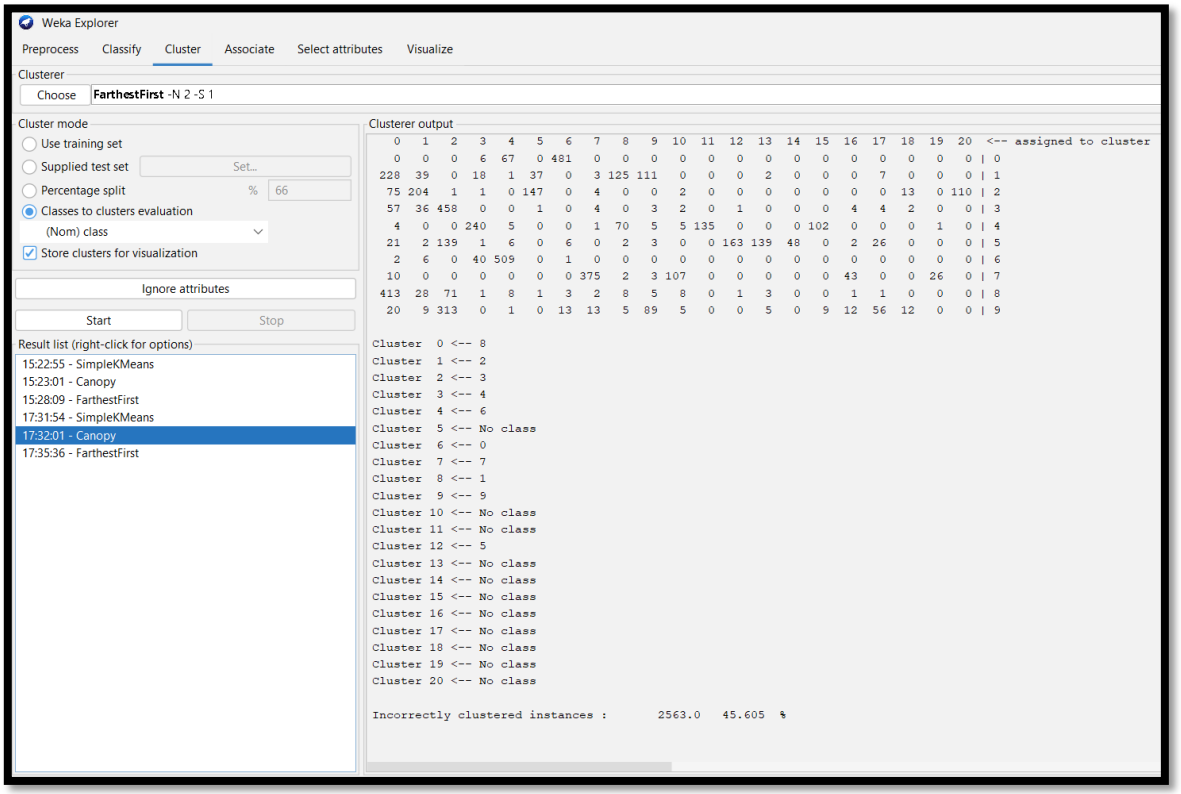


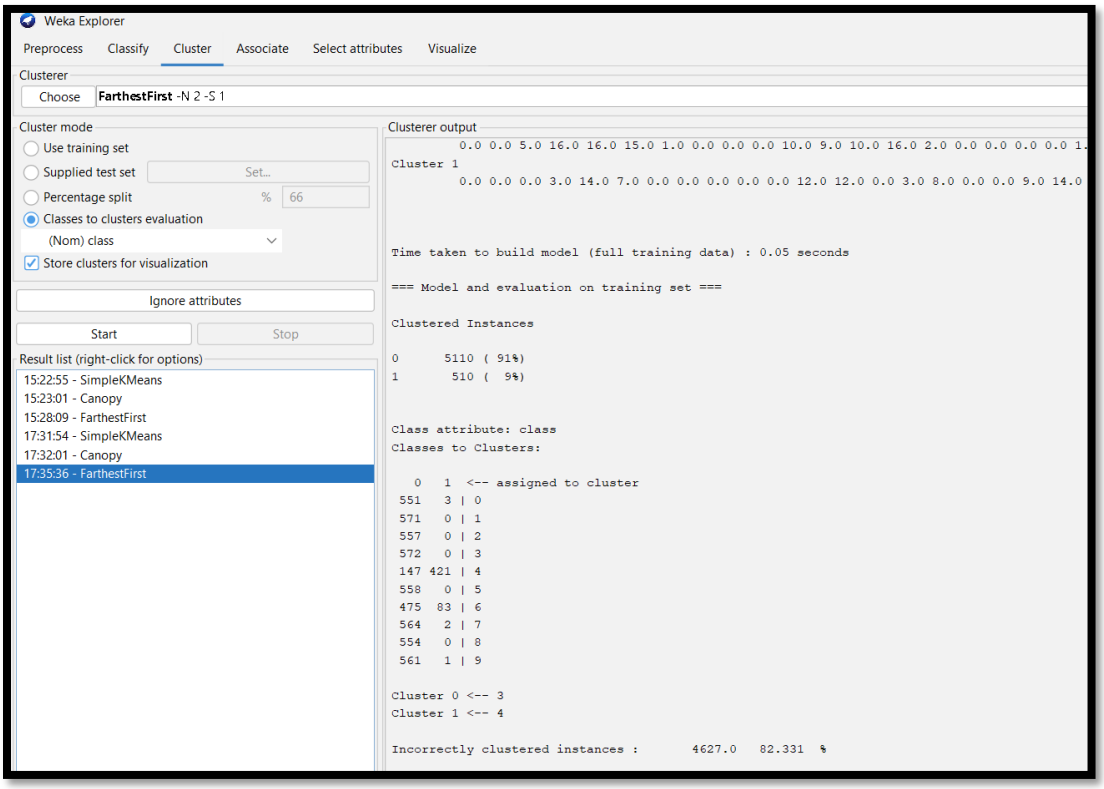



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
FarthestFirst -N 2 -S 1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:22:55 - SimpleKMeans
15:23:01 - Canopy
15:28:09 - FarthestFirst
Clusterer output
Cluster 1
Time taken to build model (full training data) : 0.1 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 5110 (91%)
1 510 (9%)
Class attribute: class
Classes to Clusters:
0 1 <-- assigned to cluster
551 3 | 0
571 0 | 1
557 0 | 2
572 0 | 3
147 421 | 4
558 0 | 5
475 83 | 6
564 2 | 7
554 0 | 8
561 1 | 9
Cluster 0 <-- 3
Cluster 1 <-- 4
Incorrectly clustered instances : 4627.0 82.331 %

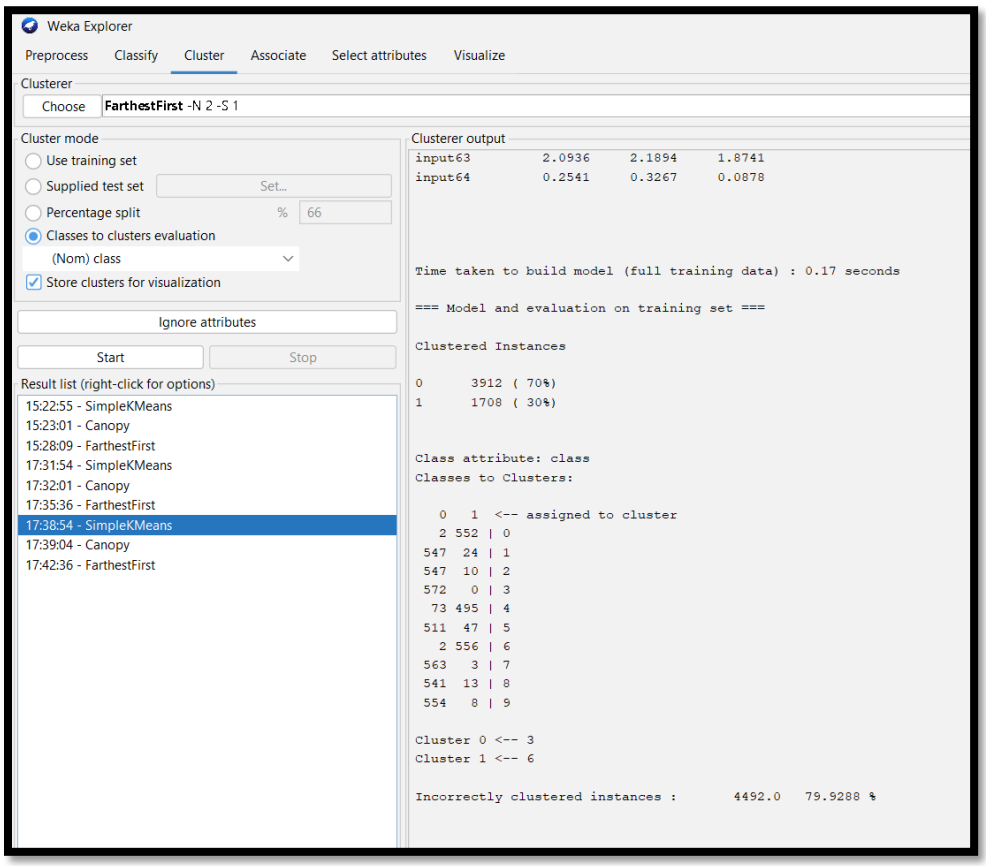
optdigits_CorrelationAttributeEval.arff

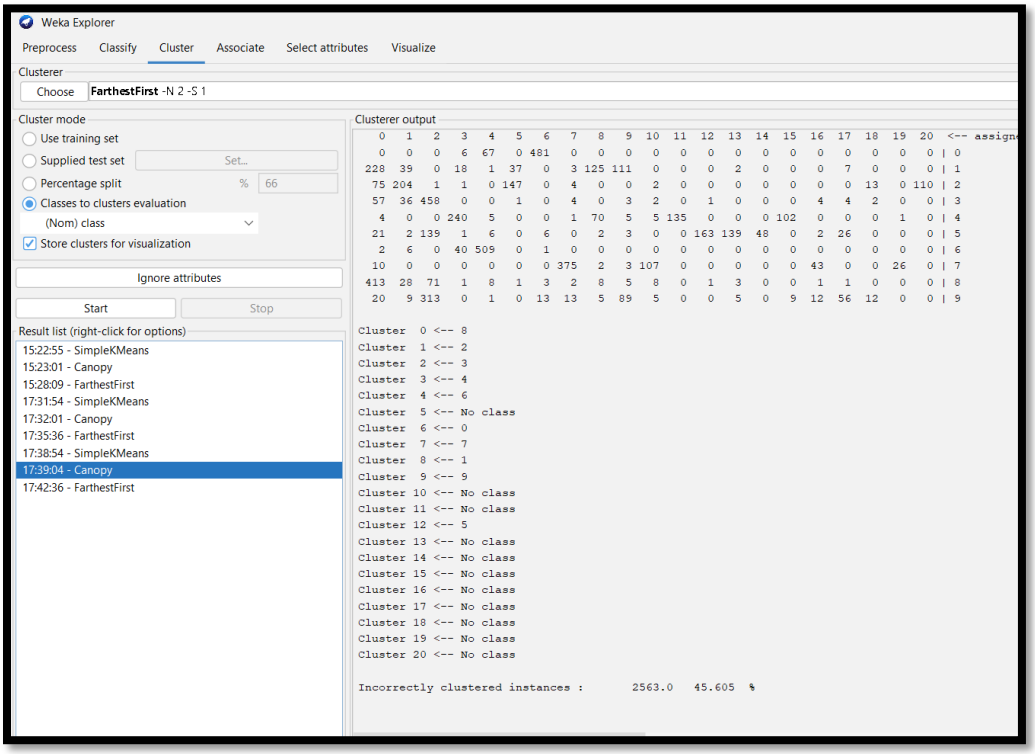


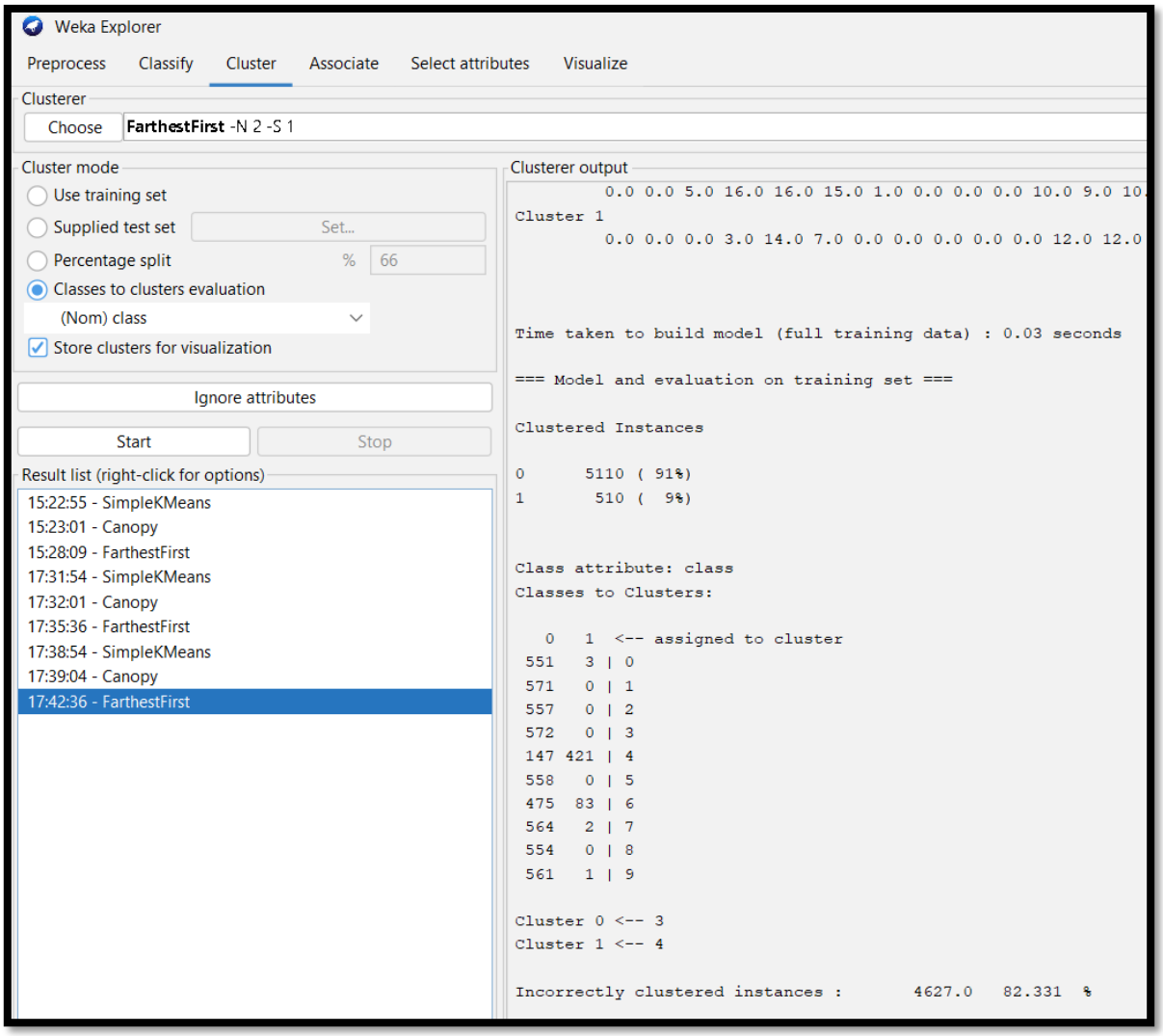



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
FarthestFirst -N 2 -S 1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:22:55 - SimpleKMeans
15:23:01 - Canopy
15:28:09 - FarthestFirst
17:31:54 - SimpleKMeans
17:32:01 - Canopy
17:35:36 - FarthestFirst
Clusterer output
0.0 0.0 5.0 16.0 16.0 15.0 1.0 0.0 0.0 0.0 10.0 9.0 10.0 16.0 2.0 0.0 0.0 0.0 0.0 1.
Cluster 1
0.0 0.0 0.0 3.0 14.0 7.0 0.0 0.0 0.0 0.0 0.0 12.0 12.0 0.0 3.0 8.0 0.0 0.0 9.0 14.0
Time taken to build model (full training data) : 0.05 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 5110 (91%)
1 510 (9%)
Class attribute: class
Classes to Clusters:
0 1 <-- assigned to cluster
551 3 | 0
571 0 | 1
557 0 | 2
572 0 | 3
147 421 | 4
558 0 | 5
475 83 | 6
564 2 | 7
554 0 | 8
561 1 | 9
Cluster 0 <-- 3
Cluster 1 <-- 4
Incorrectly clustered instances : 4627.0 82.331 %

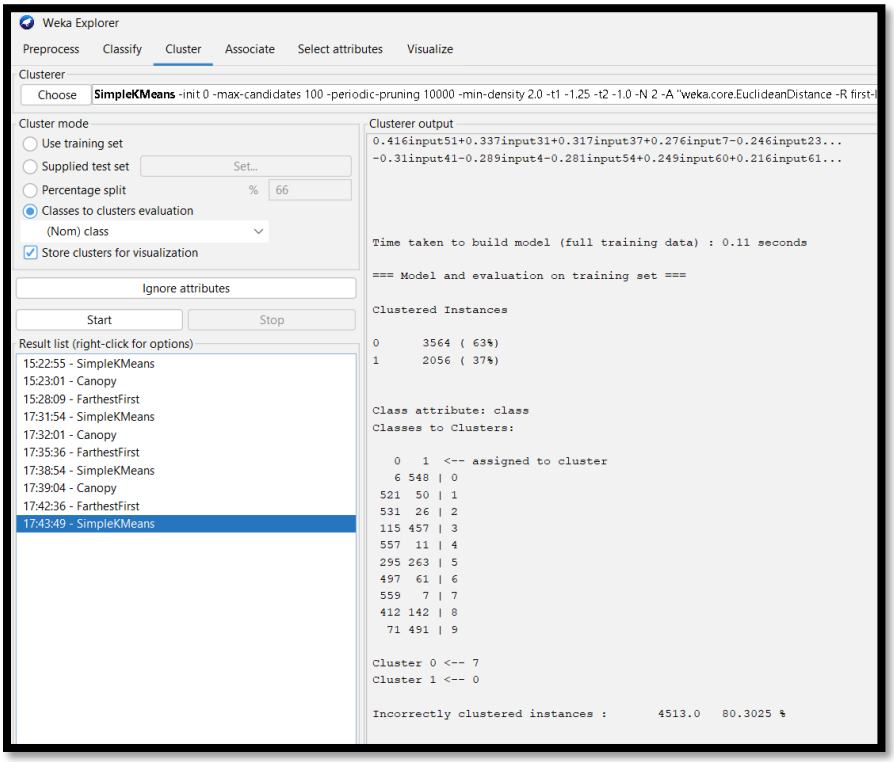
optdigits_GainRatioAttributeEval.arff

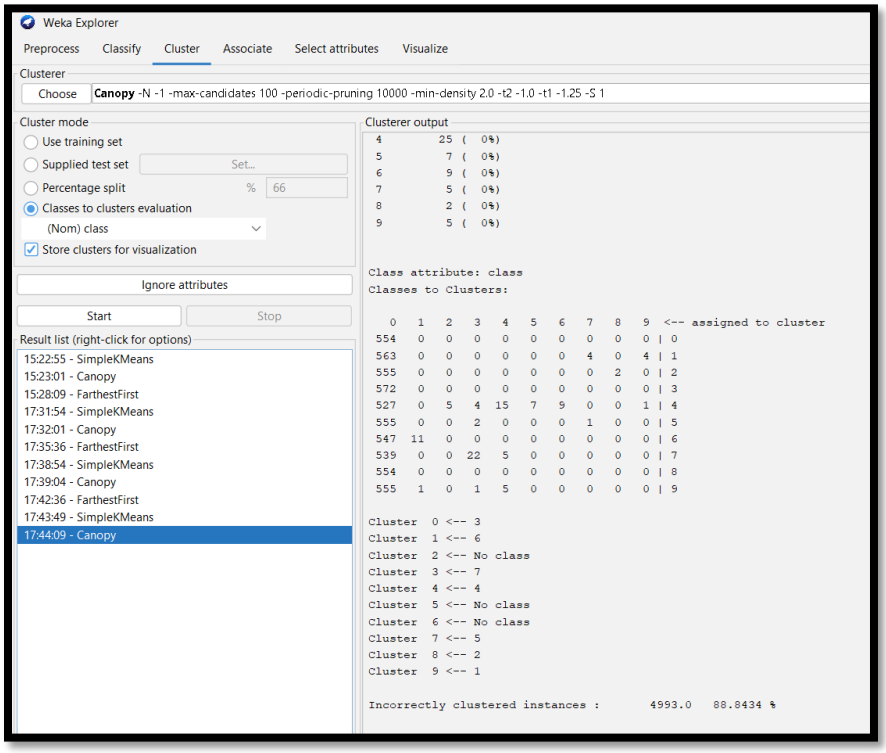


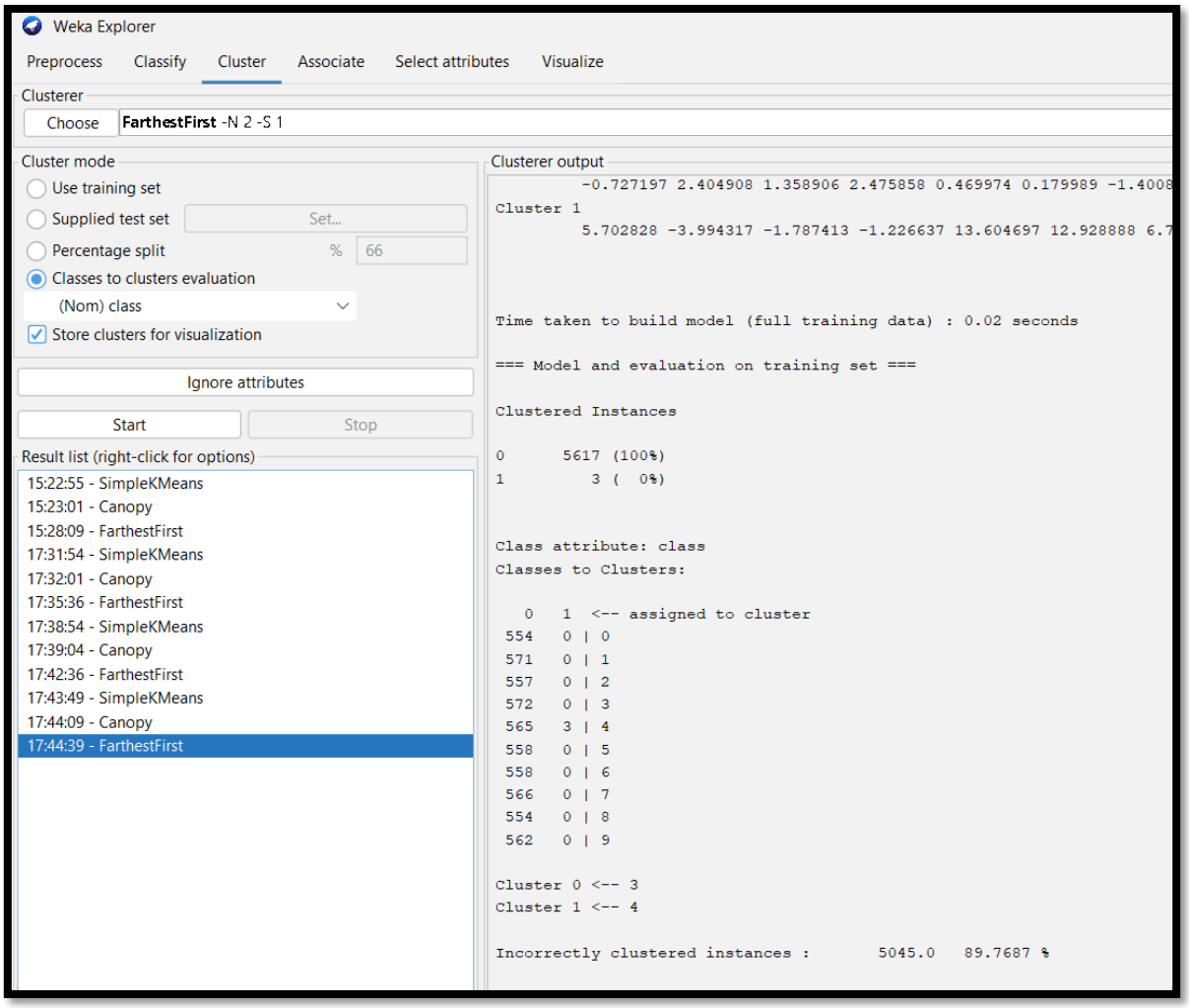



Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
FarthestFirst -N 2 -S 1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:22:55 - SimpleKMeans
15:23:01 - Canopy
15:28:09 - FarthestFirst
17:31:54 - SimpleKMeans
17:32:01 - Canopy
17:35:36 - FarthestFirst
17:38:54 - SimpleKMeans
17:39:04 - Canopy
17:42:36 - FarthestFirst
Clusterer output
0.0 0.0 5.0 16.0 16.0 15.0 1.0 0.0 0.0 0.0 10.0 9.0 10.
Cluster 1
0.0 0.0 0.0 3.0 14.0 7.0 0.0 0.0 0.0 0.0 0.0 12.0 12.0
Time taken to build model (full training data) : 0.03 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 5110 (91%)
1 510 (9%)
Class attribute: class
Classes to Clusters:
0 1 <-- assigned to cluster
551 3 | 0
571 0 | 1
557 0 | 2
572 0 | 3
147 421 | 4
558 0 | 5
475 83 | 6
564 2 | 7
554 0 | 8
561 1 | 9
Cluster 0 <-- 3
Cluster 1 <-- 4
Incorrectly clustered instances : 4627.0 82.331 %

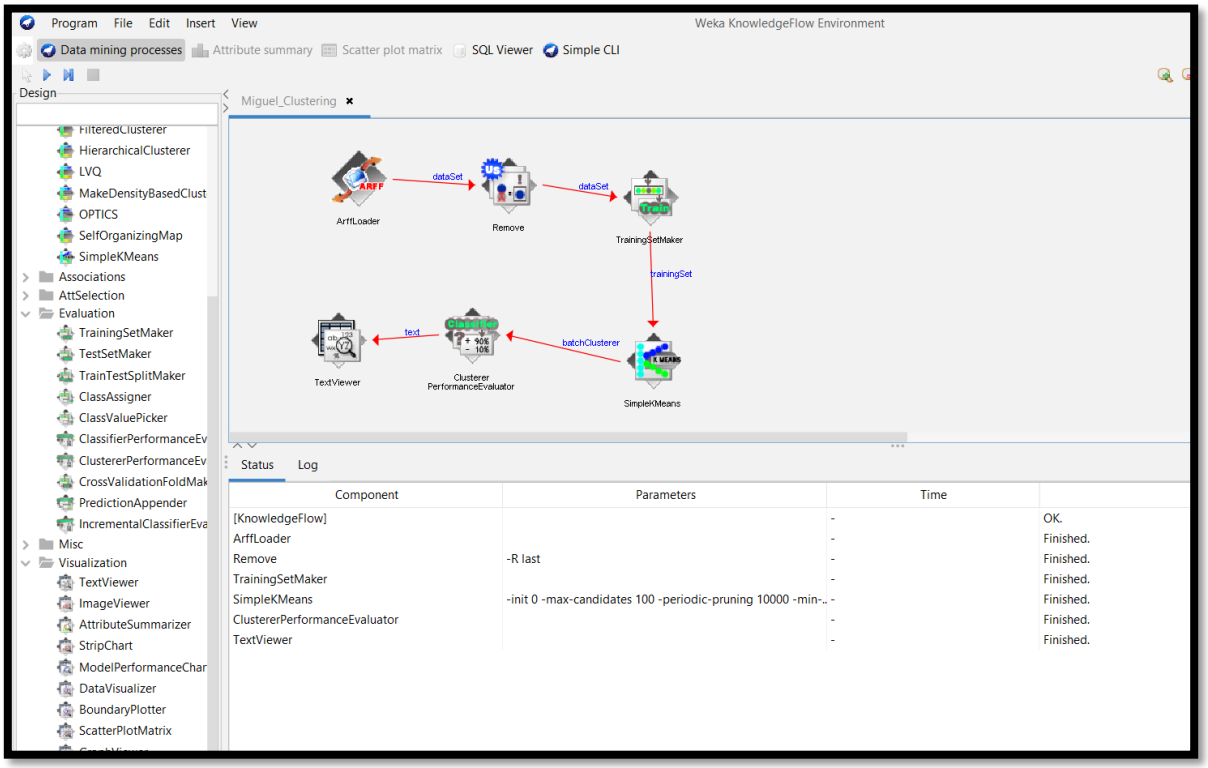
optdigits_Principal Components_v2.arff






Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
FarthestFirst -N 2 -S 1
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:22:55 - SimpleKMeans
15:23:01 - Canopy
15:28:09 - FarthestFirst
17:31:54 - SimpleKMeans
17:32:01 - Canopy
17:35:36 - FarthestFirst
17:38:54 - SimpleKMeans
17:39:04 - Canopy
17:42:36 - FarthestFirst
17:43:49 - SimpleKMeans
17:44:09 - Canopy
17:44:39 - FarthestFirst
Clusterer output
-0.727197 2.404908 1.358906 2.475858 0.469974 0.179989 -1.4008
Cluster 1
5.702828 -3.994317 -1.787413 -1.226637 13.604697 12.928888 6.7
Time taken to build model (full training data) : 0.02 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 5617 (100%)
1 3 (0%)
Class attribute: class
Classes to Clusters:
0 1 <-- assigned to cluster
554 0 | 0
571 0 | 1
557 0 | 2
572 0 | 3
565 3 | 4
558 0 | 5
558 0 | 6
566 0 | 7
554 0 | 8
562 0 | 9
Cluster 0 <-- 3
Cluster 1 <-- 4
Incorrectly clustered instances : 5045.0 89.7687 %

knowledgeFlow.



Conclusiones

La realización de este proyecto permitió consolidar el conocimiento práctico sobre el flujo de descubrimiento de conocimiento en bases de datos, destacando la importancia de la validación estadística en la toma de decisiones.

A partir de la experimentación con la base de datos optdigits, se derivan las siguientes conclusiones técnicas:

1. **Efectividad del Modelo IBk:** El análisis comparativo confirmó que el algoritmo basado en instancias (IBk) es significativamente superior (con una precisión de ~98.7%) a los enfoques basados en árboles (J48) o probabilidades (NaiveBayes) para problemas de reconocimiento de patrones visuales digitalizados.

2. **Selección vs. Transformación:** Se evidenció que las técnicas de *selección* de atributos (Correlation y GainRatio) fueron más exitosas que la *transformación* (PCA). Las primeras lograron mantener la alta precisión del modelo eliminando datos redundantes, mientras que PCA ocasionó una pérdida de información espacial relevante para la clasificación de los dígitos.

3. **Importancia de la Validación Estadística:** El uso del módulo Experimenter fue crucial para determinar que las diferencias de rendimiento observadas no eran aleatorias. Esto valida que la elección del algoritmo correcto tiene un impacto matemático real en el resultado final.

4. **Agrupamiento Autónomo:** Finalmente, mediante el flujo de datos en KnowledgeFlow, se verificó la capacidad de los algoritmos de clustering (como K-Means) para identificar correctamente la estructura de los 10 dígitos manuscritos sin necesidad de supervisión humana, validando la consistencia interna de la base de datos.

En conclusión, se determinó que la combinación óptima para este problema consiste en utilizar una técnica de selección de atributos (como CorrelationAttributeEval) junto con el clasificador IBk, logrando un equilibrio ideal entre precisión y eficiencia computacional.

Bibliografía

---

1. **Witten, I. H., Frank, E., & Hall, M. A. (2011).** *Data Mining: Practical Machine Learning Tools and Techniques* (3ra ed.). Morgan Kaufmann. *(Este es el libro "biblia" de Weka, cubre el funcionamiento del software, el Explorer, el Experimenter y algoritmos como NaiveBayes y J48).*

2. **Dua, D. & Graff, C. (2019).** *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. Recuperado de [http://archive.ics.uci.edu/ml]. *(Referencia obligatoria por haber usado la base de datos optdigits que proviene de este repositorio).*

3. **Han, J., Kamber, M., & Pei, J. (2011).** *Data Mining: Concepts and Techniques* (3ra ed.). Elsevier. *(Referencia teórica fundamental para definir los conceptos de Minería de Datos, Clasificación, Clustering y reducción de dimensionalidad incluidos en la introducción).*

4. **Quinlan, J. R. (1993).** *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers. *(Referencia específica para el algoritmo J48, que es la implementación en Java del algoritmo C4.5 descrito en este libro).*

5. **The University of Waikato. (2024).** *Weka 3: Data Mining Software in Java (Documentation)*. Recuperado de [https://www.cs.waikato.ac.nz/ml/weka/documentation.html] *(Fuente oficial para el uso de las interfaces gráficas KnowledgeFlow y los filtros de preprocesamiento).*