# Implementación de un Ensamble Básico de Clasificadores para el Pre-Procesamiento, Extracción de Conocimiento y Visualización Dentro del Proceso KDD.

*Emiliano Carrillo Moncayo* *12/junio/2020*

# Índice

# Introducción

El término Extracción de Conocimiento en Bases de Datos, o KDD por sus siglas en inglés, se refiere al amplio proceso de búsqueda de conocimiento en los datos, y ocupa métodos de minería de datos de alto nivel. Este proceso es de interés para los investigadores y expertos en campos como el Aprendizaje Automático, Reconocimiento de Patrones, Bases de Datos, Estadísticas, Inteligencia Artificial, Sistemas Expertos y Visualización de Datos. (Fayyad, 1996)

El objetivo principal del proceso KDD es extraer conocimiento de los datos en el contexto de grandes bases de datos.

Éste lo hace mediante el uso de métodos de minería de datos (algoritmos) para extraer lo que considera por conocimiento, de acuerdo con las especificaciones y medidas los expertos, utilizando una base de datos junto con cualquier método de preprocesamiento, submuestreo y transformaciones necesarios de la base de datos.

Los pasos que abarca el proceso KDD son: Limpieza de Datos, Integración de Datos, Selección de Datos, Transformación de Datos, Minería de Datos, Evaluación de Patrones, y la Representación del Conocimiento o Visualización de Datos. (Kamber et. al, 2011).

A partir de estas premisas la intención de este proyecto es implementar una aplicación de análisis de conocimiento al estilo de *Weka* y *Orange,* con la finalidad de que el desarrollador experimente dentro del contexto de las fases de Pre-procesamiento, Extracción de Conocimiento, y Visualización de Datos dentro del proceso KDD. De igual forma, obtener al final una aplicación funcional y completamente utilizable para llevar a cabo éstos pasos del proceso KDD con cualquier conjunto de Datos.

# Problema a Resolver

Se requiere implementar una aplicación básica que permita hacer análisis de Datos mediante el proceso KDD en sus primeras etapas: Pre-procesamiento, Extracción de Conocimiento, y Visualización de Datos. Utilizando un ensamble de al menos tres clasificadores comunes para clasificación de Datos. Además, permitir que sea usa usable para el usuario y se pueda ejecutar con distintos archivos de entrada de datos.

Realizar pre procesamiento de datos considerando el método de PCA para la reducción de dimensionalidad y la extracción de conocimiento previo a la clasificación. De igual forma, realizar la clasificación con el ensamble utilizando el método de Validación Cruzada para garantizar la independencia entre los datos de entrenamiento y prueba. Por último, ocupar distintas técnicas de visualización de datos que permitan al usuario extraer conocimiento otras haber realizado las predicciones correspondientes en el ensamble.

# Descripción del Proyecto Desarrollado

La aplicación implementada permite al usuario cargar cualquier conjunto de datos y llevar a cabo las primeras etapas del proceso KDD con éxito para extraer conocimiento.

Se decidió desarrollar la aplicación en el lenguaje de programación Python dentro del ambiente de desarrollo de Jupyter por la naturaleza de exploración del proyecto.

Para lograr el objetivo, la implementación consta de distintas fases: Pre-procesamiento del archivo, estandarización de datos, recibirparámetros de entrada del usuario, aplicar Principal Component Análisis tanto para la extracción de conocimiento previo a la clasificación de datos, como para reducir la dimensionalidad de los mismos y hacer más óptima la clasificación, detección de outliers mediante visualización de datos, generación del ensamble a través de 3 clasificadores: K Nearest Neighbors, Naive

Bayes, y una Red Neuronal, aplicación de Validación Cruzada para asegurar la independencia entre los datos de entrenamiento y prueba, visualización de los resultados de clasificación parcial (Por cada pliegue y modelo de clasificador) así como también del ensamble total, visualización de los límites de clasificación y matrices de confusión, y por último visualizar la dispersión de la clasificación obtenida respecto a las clases reales contra las predichas por el ensamble.

Las distintas formas de visualización de datos se implementaron para hacer más efectiva la extracción de conocimiento del Ensamble clasificador.

# Experimentos

PCA es un algoritmo de reducción de dimensionalidad no supervisado. En este ejemplo lo emplearemos para satisfacer 3 objetivos:

- **Hacer un análisis**
  Analizaremos la cantidad de componentes principales que son necesarios para resumir nuestros datos de una manera que no haya tanta pérdida de información. Este objetivo se establece únicamente como fase de experimentación.

- **Facilitar la visualización de nuestros datos**
  Ocuparemos 2 componentes principales para poder graficar nuestros datos multidimensionales en una gráfica de dispersión. Así podremos entender mejor nuestra data y podemos detectar posibles outliers.

- **Como paso de preprocesamiento para la clasificación de elementos**
  Por último ocuparemos PCA para resumir nuestros datos y eficientizar el entrenamiento de nuestro clasificador sin perder efectividad de clasificación.

## Importación de Datos

El primer paso es importar los datos que ocuparemos para el análisis. El archivo de entrada debe ser un archivo de texto plano con el formato siguiente:

```
No. Elementos
No. Atributos
No. Clases
atrib_0, atrib_1, ..., atrib_n, clase
atrib_0, atrib_1, ..., atrib_n, clase
... ... ...
atrib_0, atrib_1, ..., atrib_n, clase
```

## Preprocesamiento del Archivo Base

Primero preprocesamos el archivo para obtener los metadatos de No. de elementos, atributos y clases que éste contiene en el encabezado y así construir nuestro dataset.

Podemos obtener un pequeño vistazo de cómo se ve nuestro dataset hasta ahora.

| | atrib_1 | atrib_2 | atrib_3 | atrib_4 | atrib_5 | atrib_6 | atrib_7 | atrib_8 | atrib_9 | atrib_10 | atrib_11 | atrib_12 | atrib_13 | atrib_14 | atrib_15 | atrib_16 | atrib_17 | atrib_18 | atrib_19 | clase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 140 | 125 | 0 | 0.0 | 0.0 | 0.277778 | 0.062963 | 0.666667 | 0.311111 | 6.185185 | 7.333334 | 7.666666 | 3.555556 | 3.444444 | 4.444445 | -7.888889 | 7.777778 | 0.545635 | -1.121818 | 0 |
| 1 | 188 | 133 | 0 | 0.0 | 0.0 | 0.333333 | 0.266667 | 0.500000 | 0.077778 | 6.666666 | 8.333334 | 7.777778 | 3.888889 | 5.000000 | 3.333333 | -8.333333 | 8.444445 | 0.538580 | -0.924817 | 0 |
| 2 | 105 | 139 | 0 | 0.0 | 0.0 | 0.277778 | 0.107407 | 0.833333 | 0.522222 | 6.111111 | 7.555555 | 7.222222 | 3.555556 | 4.333334 | 3.333333 | -7.666666 | 7.555555 | 0.532628 | -0.965946 | 0 |
| 3 | 34 | 137 | 0 | 0.0 | 0.0 | 0.500000 | 0.166667 | 1.111111 | 0.474074 | 5.851852 | 7.777778 | 6.444445 | 3.333333 | 5.777778 | 1.777778 | -7.555555 | 7.777778 | 0.573633 | -0.744272 | 0 |
| 4 | 39 | 111 | 0 | 0.0 | 0.0 | 0.722222 | 0.374074 | 0.888889 | 0.429629 | 6.037037 | 7.000000 | 7.666666 | 3.444444 | 2.888889 | 4.888889 | -7.777778 | 7.888889 | 0.562919 | -1.175773 | 0 |

**Dataset original**

## Preprocesamiento de  Datos

Como PCA se soporta de la desviación estándar de los datos para calcular la nueva proyección de nuestros datos, una variable con una desviación estándar alta tendrá un peso mayor para el cálculo de la proyección que una variable con una desviación estándar baja. Si normalizamos los datos, todas las variables tendrán la misma desviación estándar, por lo tanto, el cálculo no estará cargado.

Además, como no tenemos conocimiento del dominio del conjunto de datos de ejemplo, no sabemos si las unidades de medida de sus variables son distintas. Otra razón por la cual normalizar nuestros datos.

PCA se considera como un algoritmo no supervisado, esto quiere decir que se apoya únicamente del set de datos sin las clases asignadas. Por esto, el primer paso de preprocesamiento será dividir nuestro set en dos: el set con los atributos y el set de las clases de asignación. Paso continuo sería estandarizar los datos sin la columna de las clases.

| | atrib_1 | atrib_2 | atrib_3 | atrib_4 | atrib_5 | atrib_6 | atrib_7 | atrib_8 | atrib_9 | atrib_10 | atrib_11 | atrib_12 | atrib_13 | atrib_14 | atrib_15 | atrib_16 | atrib_17 | atrib_18 | atrib_19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.058049 | -0.285629 | 0.0 | -0.338097 | -0.199868 | -0.621918 | -0.110554 | -0.463761 | -0.090389 | -0.784836 | -0.693751 | -0.796105 | -0.848481 | 1.511535 | -0.712136 | -0.070207 | -0.849913 | 0.757935 | -0.001032 |
| 1 | 0.598297 | -0.153891 | 0.0 | -0.338097 | -0.199868 | -0.598559 | -0.103935 | -0.509463 | -0.093776 | -0.769504 | -0.659068 | -0.793056 | -0.837138 | 1.680258 | -0.772656 | -0.106443 | -0.831181 | 0.724362 | 0.119387 |
| 2 | -0.536634 | -0.055087 | 0.0 | -0.338097 | -0.199868 | -0.621918 | -0.109110 | -0.418060 | -0.087325 | -0.787194 | -0.686044 | -0.808304 | -0.848481 | 1.607948 | -0.772656 | -0.052089 | -0.856157 | 0.696035 | 0.094247 |
| 3 | -1.507478 | -0.088021 | 0.0 | -0.338097 | -0.199868 | -0.528480 | -0.107184 | -0.341891 | -0.088024 | -0.795450 | -0.678336 | -0.829651 | -0.856043 | 1.764620 | -0.857382 | -0.043030 | -0.849913 | 0.891177 | 0.229748 |
| 4 | -1.439109 | -0.516170 | 0.0 | -0.338097 | -0.199868 | -0.435042 | -0.100444 | -0.402826 | -0.088669 | -0.789553 | -0.705312 | -0.796105 | -0.852262 | 1.451277 | -0.687929 | -0.061148 | -0.846791 | 0.840188 | -0.034012 |

**Set de atributos**

## Aplicación del PCA

A continuación aplicaremos el PCA con tantos componentes principales como especifique el usuario. Se calculan 2 por defecto para que a continuación podamos graficar nuestros datos.

```
nComponentes = int(input())
pca = PCA(n_components=nComponentes)
```

```
2
```

Input de no. de componentes

| | PC1 | PC2 |
|---|---|---|
| 0 | -2.299677 | -0.343837 |
| 1 | -2.371925 | -0.403867 |
| 2 | -2.389590 | -0.266572 |
| 3 | -2.514506 | -0.096834 |
| 4 | -2.236746 | -0.124341 |

Dataset después de aplicar poca con 2 componentes principales

## Análisis de resultados del PCA

A continuación calcularemos la razón de varianza para cada componente. Ésta relación es la varianza causada por cada componente princiapl. Esta nos sirve para observar que tan bien los componentes principales calculados representan a nuestros datos originales.
Estos son las razones para los 19 componentes principales.

```
[4.13814900e-01, 1.65640288e-01, 1.06345188e-01, 6.27965215e-02,
5.78674248e-02, 4.70498598e-02, 4.20205907e-02, 3.83644353e-02,
2.77241028e-02, 1.90903343e-02, 1.22596944e-02, 4.70502749e-03,
2.29038862e-03, 3.12439197e-05, 1.49188547e-16, 1.03612940e-16,
9.12790822e-17, 7.73403277e-17, 9.78771803e-34]
```

Razones para los 19 componentes

Podemos observar que el CP1 es responsable de 41.3% de la varianza. Similarmente, el CP2 causa el 16.5% de la varianza en nuestro set de datos. Por lotanto, podemos decir que colectivamente, los dos primeros componentes principales capturan el 57.8% (41.3% + 16.5%) de la información de nuestro dataset, lo cual no es tan óptimo pero nos puede ser útil, por ejemplo, para graficar nuestros datos.

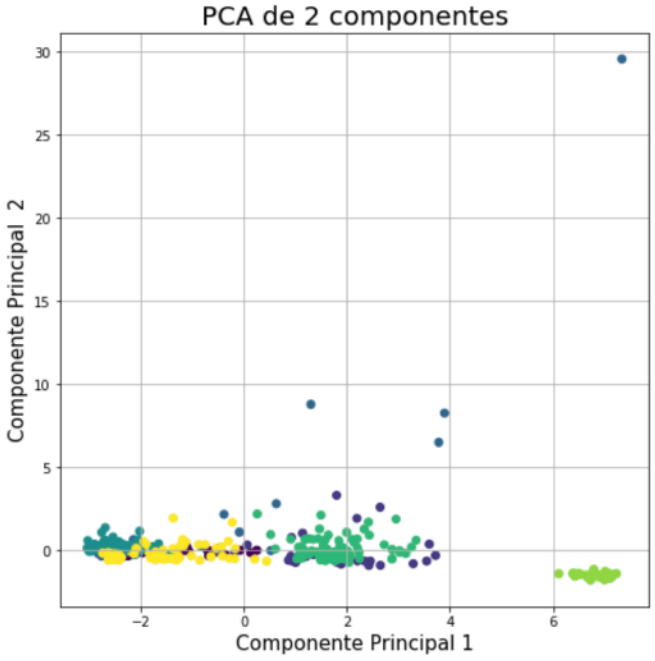
Como ejemplo, si quisieramos representar un 85% de nuestros datos, tendríamos que ocupar los 6 Componentes picipales primarios.

```
PC1: 0.414
PC2: 0.166
Total: 57.95%
```

**Explained_variance compuesta del PC1 y**

## Visualización del PCA en 2 Dimensiones

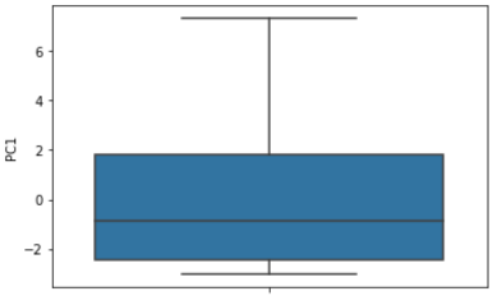
Nuestro dataset original contenía 19 dimensiones las cuales, a través del PCA, logramos reducir a 2. Esto lo hicimos para poder graficar nuesta data y poder recuperar patrones



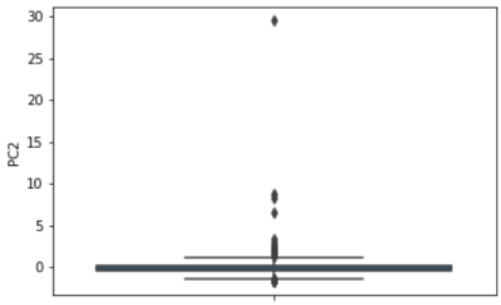

**Gráfica de dispersión de los datos en dos**

## Detección de Outliers

Al graficar observamos que hay elementos en nuestro dataset que se comportan de manera extraña y están muy alejados de los demás.
Se puede ver mucha varianza en el componente 2, con algunos elementos muy alejados de la media. Éstos son posibles elementos anómalos o outliers que, sin el dominio del dataset, no podríamos evaluar con certeza.
Primero, nos apoyaremos de la técnica de graficación por bigotes para detectar outliers en los dos componentes.
En el PC 1 no vemos elementos graficados fuera de los bigotes, lo que nos indica que no hay casos anómalos.
A diferencia del PC1, en el PC2 podemos observar que existen elementos muy alejados de la media y de los bordes superior e inferior (Q1 y Q3).



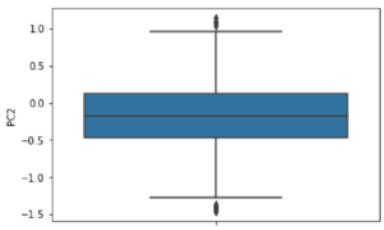

Gráfica de bigotes para el PC1



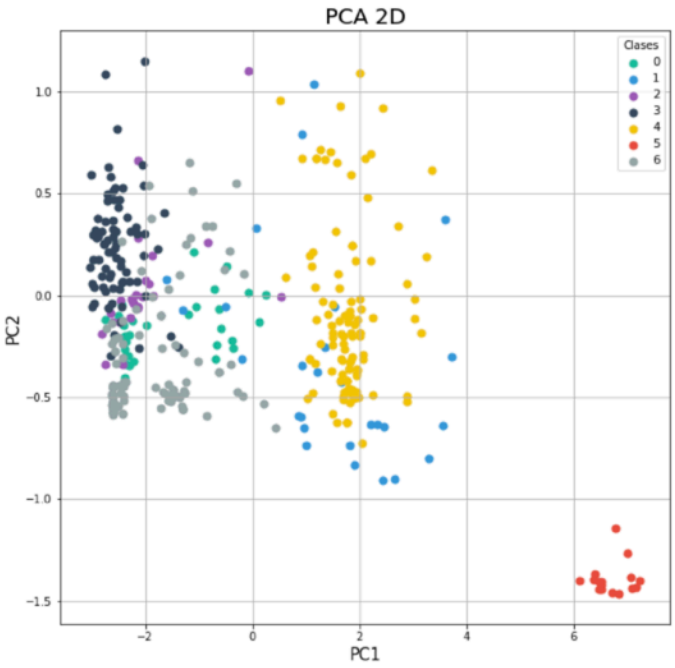

Gráfica de bigotes para el PC2

Para efectos de la experimentación, supondremos que sí son outliers y queremos recortarlos. Para esto nos apoyaremos del método del Rango Inter-Cuartil (IQR) que es el mismo que emplean las gráficas de bigotes para su graficación. Este método observa la dispersión estadística de nuestros datos. Con esto podremos detectar aquellos elementos que están muy alejados de la media y sobrepasan los límites superior e inferior, y eliminarlos.

Calculamos los límites inferior y superior. Una vez identificados los límites podemos observar aquellos elementos que los exceden. Una vez identificados nuestros outliers, procederemos a eliminarlos.




Gráfica de bigotes para el PC2




**Gráfica de dispersión De los datos en 2d**
**después de recortar outliers**

Una vez que limpiamos nuestros datos, procederemos a graficar de nuevo nuestros dos primeros componentes principales para observar de mejor manera como se dispersan nuestras clases.

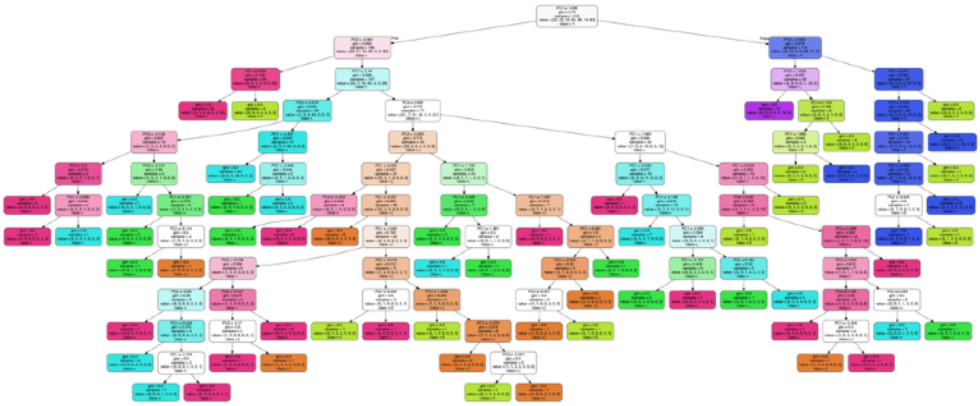
## Clasificación

Tener una gran cantidad de atributos en un dataset afecta el rendimiento y la precisión de los algoritmos de clasificación. Nuestro dataset original contenía 19 atributos, los cuales, a través de la técnica de reducción de dimensionalidad de PCA, logramos reducir a 2.

En este ejemplo entrenaremos un árbol de decisión con nuestros datos reducidos con PCA. En seguida, analizaremos la precisión de éste cuando es entrenado con distintas cantidades de Componentes Principales. El objetivo es ver el número óptimo de Componentes Principales que nos permitan reducir el tiempo de entrenamiento del clasificador al resumir nuestros datos adecuadamente, y conservar un elevado porcentaje de precisión.

El método de clasificación por árbol de decisión es un método de aprendizaje supervisado, por esto, debemos entrenarlo con el set de atributos y su clasificación inicial. Además, para probar la precisión de éste, necesitamos un set de prueba. Por ello procederemos a partir nuestro set de datos en 2 secciones: Un set para entrenar a nuestro clasificador, y uno para entrenarlo.

Procederemos a entrenar nuestro clasificador con el set de datos de entrenamiento y sus respectivas clases. A continuación se muestra el árbol de decisión generado. El objetivo es generar un árbol no tan profundo para que la toma de decisiónes sea rápida.



**Árbol de decisión generado tras entrenar clasificador con los**

## Predicción y Análisis

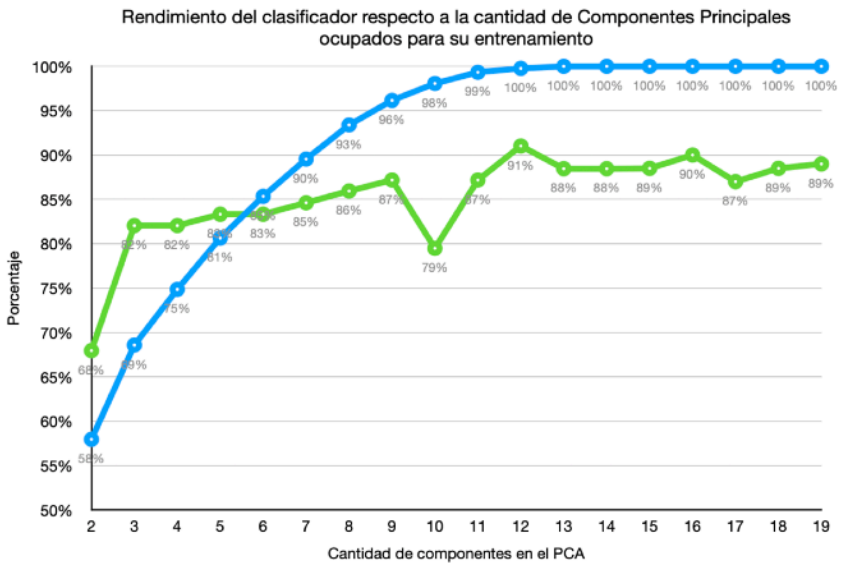
Por último predeciremos la clasificación de los datos de prueba que apartamos del set original antes de clasificar y compararemos, a través de una matriz de confusión, que tanto éstos se alejan de su verdadera clasificación.

Podemos obeservar que al ocupar 2 componentes principales, estamos representando el 57.95% de nuestros datos originales y la precisión de nuestro clasificador es de 70.51%, lo cual no es óptimo.

Para observar la variación en la precisión del clasificador dependiendo de la cantidad de componentes principales que ocupemos realizamos múltiples experimentos, entrenando el clasificador el resultado del PCA tras alterar la cantidad de componentes principales que deseabamos.

A continuación se presenta la gráfica que relaciona la cantidad de componentes principales ocupados para el entrenmamiento del clasificador y su respectiva calificación de precisión de clasificación.

Como se observa en la gráfica, a partir de los 11-12 componentees principales, se tiene una casi perfecta representación de los datos principales. A su vez, es en estos valores que el clasificador tiene su máximo en cuanto a grado de precisión. Esto quiere decir que reducir nuestra data a 12 dimensiones, en este ejmplo, sería óptimo para el clasificador.




**Métricas y matriz de confusión tras probar nuestro**

```
Número de Componentes Principales ocupados:
2
Explained_Variance:
57.95%

Matriz de confusión:
[[ 4  0  1  1  0  0  2]
 [ 0  1  0  0  3  0  0]
 [ 2  0  3  0  0  0  3]
 [ 0  0  0 11  0  0  2]
 [ 0  4  0  0 21  0  0]
 [ 0  0  0  0  0  2  0]
 [ 0  2  2  0  0  0 14]]

Precición de clasificación:
71.79%
```

## Experimentos y Desarrollo con Clasificadores

El siguiente experimento tuvo como objetivo escoger y experimentar con los 3 modelos de clasificadores que ocuparé para mi ensamble final. Para esto, se ocuparon las técnicas de preprocesamiento implementadas en la sección pasada. Objetivamente, la extracción de variables y procesamiento del archivo de texto, la estandarización de datos por medio del método StandardScaler, y la aplicación del PCA.

Se tomó la decisión de ocupar PCA como método para cada modelo de clasificador ya que, tras experimentar con esto, se observó que optimizaba todos los casos. De igual forma empecé a experimentar con el nuevo dataset con 385 atributos. Para este dataset en específico observé que la cantidad de componentes óptima para el PCA eran 12 así que la experimentación a continuación muestra los resultados de los clasificadores tras reducir la dimensionalidad del dataset a 12 componentes principales.

Decidí ocupar los siguientes métodos de clasificación supervisados para hacer mi experimentación: K-Nearest-Neighbors, Naive Bayes, y una Red Neuronal.
Para todos los experimentos expuestos a continuación se dividió el dataset en sets de entrenamiento y pruebas, y se ocuparon los mismos sets para entrenar y probar cada uno de éstos.

## K-Nearest-Neighbors

Para el modelo KNN experimenté con el parámetro de cantidad de vecinos (valor K) para ver cuál valor era el más apropiado para este dataset en específico. Para ello construí un loop que me calculó el error promedio tras hacer la evaluación para cada valor k de 1 a 40. La gráfica es la siguiente:

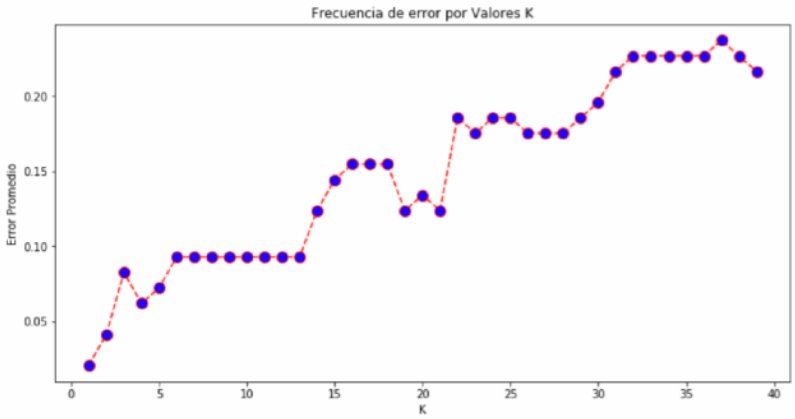



FIG 1. FRECUENCIA DE ERROR PARA DISTINTOS VALORES K

Tras varios experimentos observé que con 5 vecinos cercanos obtenía resultados de predicción más elevados. Al final obtuve un puntaje de predicción del 93% con KNN.

```
******* CLASIFICADOR KNN *******

Matriz de confusión:
[[11  1  0  0  0  0]
 [ 0 40  0  0  0  1]
 [ 0  1  9  0  2  0]
 [ 0  0  0  4  0  0]
 [ 0  0  0  0  6  2]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.95      0.98      0.96        41
           2       1.00      0.75      0.86        12
           3       1.00      1.00      1.00         4
           4       0.75      0.75      0.75         8
           5       0.87      1.00      0.93        20

    accuracy                           0.93        97
   macro avg       0.93      0.90      0.91        97
weighted avg       0.93      0.93      0.93        97


Puntaje de precisión:
0.9278350515463918
```

FIG 2. RESULTADOS DEL CLASIFICADOR KNN

```
******* CLASIFICADOR NAIVE BAYES *******

Matriz de confusión:
[[ 9  1  0  0  0  2]
 [ 1 39  0  0  0  1]
 [ 2  2  8  0  0  0]
 [ 0  0  0  4  0  0]
 [ 0  1  0  0  5  2]
 [ 0  1  0  0  0 19]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.75      0.75      0.75        12
           1       0.89      0.95      0.92        41
           2       1.00      0.67      0.80        12
           3       1.00      1.00      1.00         4
           4       1.00      0.62      0.77         8
           5       0.79      0.95      0.86        20

    accuracy                           0.87        97
   macro avg       0.90      0.82      0.85        97
weighted avg       0.88      0.87      0.86        97


Puntaje de precisión:
0.865979381443299
```

## Naive Bayes

Para el clasificador de Naive Bayes se ocupó el modelo gausiano por la naturaleza y forma del dataset. Originalmente estaba ocupando un algoritmo Multinominal y conseguía resultados extraños.

Esto pasó porque mientras que el algoritmo multinominal se ocupa para atributos discretos, el Gausiano espera que tu dataset siga una distribución normal. Cosa que se consiguió tras escalar los datos en la sección pasada de procesamiento.
Estos son los resultados que se obtuvieron con NaiveBayes. Mientras que Bayes tuvo un buen resultado de precisión, no fue el mejor con los parámetros que les colocamos a todos los clasificadores. Fue el que tuvo una menor taza de predicción.

FIG 3. RESULTADOS DEL CLASIFICADOR NAIVE BAYES

## Neural Network (Multi Layer Perceptron)

Para el tercer y último clasificador se ocupó el MLPClassifier (Multi Layer Perceptron) el cual es una red neuronal. Para la experimentación y obtención de los mejores parámetros del modelo se corrió un Grid Search el cual permitió probar con distintas combinaciones de parámetros y automáticamente seleccionar la mejor para el modelo.

Por último me arrojó la combinación de parámetro óptima para mi modelo al correrlo por 300 iteraciones.Los parámetros óptimos fueron:

```
{
'activation': 'tanh',

'alpha': 0.05, 'hidden_layer_sizes': (50, 50, 50),
'learning_rate': ‘constant', 'solver': ‘adam'
}
```

Los resultados obtenidos con dichos valores de parámetros me dieron un porcentaje de predicción de 98% siendo el más alto de los tres modelos seleccionados.

```
******* CLASIFICADOR NEURAL NET*******

Matriz de confusión:
[[11  1  0  0  0  0]
 [ 0 41  0  0  0  0]
 [ 0  0 12  0  0  0]
 [ 0  0  0  4  0  0]
 [ 0  0  0  0  8  0]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.98      1.00      0.99        41
           2       1.00      1.00      1.00        12
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         8
           5       1.00      1.00      1.00        20

    accuracy                           0.99        97
   macro avg       1.00      0.99      0.99        97
weighted avg       0.99      0.99      0.99        97


Puntaje de precisión:
0.9896907216494846
```

FIG 5. RESULTADOS DEL CLASIFICADOR NEURAL NET

## Implementación del Ensamble

```
******* CLASIFICADOR NEURAL NET*******

Matriz de confusión:
[[11  1  0  0  0  0]
 [ 0 41  0  0  0  0]
 [ 0  0 12  0  0  0]
 [ 0  0  0  4  0  0]
 [ 0  0  0  0  8  0]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.98      1.00      0.99        41
           2       1.00      1.00      1.00        12
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         8
           5       1.00      1.00      1.00        20

    accuracy                           0.99        97
   macro avg       1.00      0.99      0.99        97
weighted avg       0.99      0.99      0.99        97


Puntaje de precisión:
0.9896907216494846
```

Esta vez mi objetivo era comenzar a experimentar con l implementación del ensamble y la agrupación de mis tres modelos de clasificación: K Nearest Neighbors, Red Neuronal, y Naive Bayes.

Decidí ocupar como ensamble un método de máximo de votaciones. Quiere decir que tomo la moda de los resultados de cada data-point para cada uno de mis modelos. Con esto saco mi predicción final, la predicción del ensamble.

Los resultados que había obtenido con cada uno de mis clasificadores por separados los siguientes:

```
******* CLASIFICADOR KNN *******

Matriz de confusión:
[[11  1  0  0  0  0]
 [ 0 40  0  0  0  1]
 [ 0  1  9  0  2  0]
 [ 0  0  0  4  0  0]
 [ 0  0  0  0  6  2]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.95      0.98      0.96        41
           2       1.00      0.75      0.86        12
           3       1.00      1.00      1.00         4
           4       0.75      0.75      0.75         8
           5       0.87      1.00      0.93        20

    accuracy                           0.93        97
   macro avg       0.93      0.90      0.91        97
weighted avg       0.93      0.93      0.93        97


Puntaje de precisión:
0.9278350515463918
```

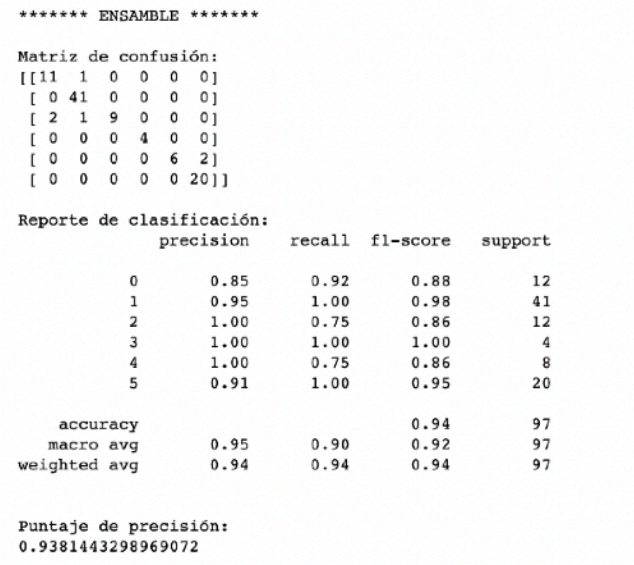
RESULTADOS OBTENIDOS CON LOS MODELOS KNN Y LA RED NEURONAL

```
******* CLASIFICADOR NAIVE BAYES *******

Matriz de confusión:
[[ 9  1  0  0  0  2]
 [ 0 40  0  0  0  1]
 [ 2  2  8  0  0  0]
 [ 0  0  0  4  0  0]
 [ 0  1  0  0  5  2]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.82      0.75      0.78        12
           1       0.91      0.98      0.94        41
           2       1.00      0.67      0.80        12
           3       1.00      1.00      1.00         4
           4       1.00      0.62      0.77         8
           5       0.80      1.00      0.89        20

    accuracy                           0.89        97
   macro avg       0.92      0.84      0.86        97
weighted avg       0.90      0.89      0.88        97


Puntaje de precisión:
0.8865979381443299
```

RESULTADOS OBTENIDOS CON NAIVE BAYES

RESULTADOS OBTENIDOS INTEGRANDO EL ENSAMBLE TOTAL

Como podemos observar la red neuronal operó mejor que el ensamble final. Deduzco que ocurre esto porque fue a la red neuronal a la única que le hice un grid search para obtener la



```
******* ENSAMBLE *******

Matriz de confusión:
[[11  1  0  0  0  0]
 [ 0 41  0  0  0  0]
 [ 2  1  9  0  0  0]
 [ 0  0  0  4  0  0]
 [ 0  0  0  0  6  2]
 [ 0  0  0  0  0 20]]

Reporte de clasificación:
              precision    recall  f1-score   support

           0       0.85      0.92      0.88        12
           1       0.95      1.00      0.98        41
           2       1.00      0.75      0.86        12
           3       1.00      1.00      1.00         4
           4       1.00      0.75      0.86         8
           5       0.91      1.00      0.95        20

    accuracy                           0.94        97
   macro avg       0.95      0.90      0.92        97
weighted avg       0.94      0.94      0.94        97


Puntaje de precisión:
0.9381443298969072
```

mejor configuración de sus parámetros de entrada.

Para la siguiente entrega planeo que mi código de la opción al usuario de buscar automáticamente la mejor configuración de parámetros para cada uno de los modelos automáticamente o éste los pueda ingresar a su disposición.

# Conclusiones

La Minería de Datos  brinda a los negocios información valiosa sobre los problemas que enfrentan y ayuda a identificar nuevas oportunidades. Además, ayuda a las empresas a resolver problemas más complejos y tomar decisiones más inteligentes. Es por esto que la Minería de Datos es una herramienta muy poderosa para las empresas; Sin embargo, se necesita de gente que la aplique y desarrolle. Es por esto que la implementación de herramientas que permitan el fácil uso del Proceso KDDes muy importante.

Al implementar herramientas que faciliten la extracción de conocimiento de Bases de Datos de instituciones, organizaciones, libres, etc. La gente experta en su dominio puede tomar mejores y más soportadas decisiones.

## *Referencias*

Fayyad, Piatetsky-Shapiro, Smyth, "From Data Mining to Knowledge Discovery: An Overview", in Fayyad, Piatetsky-Shapiro, Smyth, Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, AAAI Press / The MIT Press, Menlo Park, CA, 1996, pp.1-34

Han, J., Kamber, M., & Pei, J. (2011). *Data mining: concepts and techniques*. Amsterdam: Elsevier.

Nigro, Héctor Oscar.  INCA/INTIA - Departamento de Computación y Sistemas. “KDD (Knowledge Discovery in Databases): Un proceso centrado en el usuario”. Accedido desde: http://sedici.unlp.edu.ar/bitstream/handle/10915/21220/Documento_completo.pdf?sequence=1&isAllowed=y

Medium. 2020. *Ways To Detect And Remove The Outliers*. [online] Available at: <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba> [Accessed 13 June 2020].

Stack Abuse. 2020. *Implementing PCA In Python With Scikit-Learn*. [online] Available at: <https://stackabuse.com/implementing-pca-in-python-with-scikit-learn/> [Accessed 13 June 2020].