# PROYECTO DE INTELIGENCIA ARTIFICAL


**MANUELA GUTIÉRREZ CANO**

**DANIEL ESTEBAN MAYA PORTILLO**

**NILSON SUÁREZ HERNÁNDEZ**

**UNIVERSIDAD DE ANTIOQUIA**

**FACULTAD DE INGENIERÍA**

**INTRODUCCIÓN A LA INTELIGENCIA ARTIFICAL**

**MEDELLÍN**

**2022**

# Tabla de contenido

DESARROLLO DEL PROYECTO

# 1. Introducción

En el presente trabajo se desarrollará todos y cada una de las etapas necesarias para resolver un problema de Machine Learning o Aprendizaje Automático de tipo supervisado, ya que el problema predictivo consta de un conjunto de datos que tiene asociado una variable de salida o target que es la que se quiere predecir con ayuda de algunos modelos supervisados como Random Forest, Support Machine, Red Neuronal, Árbol de Decisión y Gradient Boosting Tree, con unos parámetros que se deben ajustar, también llamados hiper parámetros de los modelos. Antes de esto se realizará la exploración de los datos o entendimiento del problema que es fundamental para saber cuál es el comportamiento de estos y de qué manera se debe abordar el proyecto, así como también se utilizarán las curvas de aprendizaje para la interpretación de los resultados que arrojaron los distintos modelos teniendo en cuenta la métrica de evaluación para este trabajo que es el accuracy para evaluar y analizar el rendimiento de los algoritmos mencionados anteriormente.

# 2. Planteamiento del problema

Más allá de la molestia y el tiempo que se pierde a causa de los mensajes no deseados, el spam puede causar daños significativos, infectando las computadoras de los usuarios con software malicioso capaz de dañar los sistemas y robar información personal. También puede consumir recursos de la red.

Ante esta problemática se busca categorizar y predecir si el tipo de correo o mensaje pertenece a este grupo que presentan frecuencias de palabras como características, las cuales se utilizarán para el entrenamiento del método de aprendizaje supervisado.

# 3. Dataset

El dataset que se tendrá en cuenta para este tipo de clasificación se ha tomado de la plataforma Kaggle (https://www.kaqqle.com/datasets/colormap/spambase) que presenta una base de datos de 4601 de muestras y cuyos atributos son 58, de los cuales se consideran relevantes los siguientes:

- **Word_freq_address** : Porcentaje de palabras en el correo electrónico que coinciden con la dirección
- **charfreq#:** Porcentaje de caracteres en el correo electrónico que coinciden con el número.
- **capital_run_length_average:** Promedio de secuencias ininterrumpidas de letras mayúsculas.
- **capital_run_length_longest:** Longitud de la secuencia ininterrumpida más larga de letras mayúsculas.
- **capital_run_length_total:** Número total de letras mayúsculas en el correo electrónico.

Hay que tener en cuenta que para que el dataset cumpla con los requisitos del proyecto se completa el número de muestras con datos faltantes, es decir, 400 datos faltantes en el dataset, y además se crean las variables categóricas haciendo uso de la librería pandas.

## 4. Exploración descriptiva de los datos

Para realizar la exploración de los datos y lograr comprender cómo es el comportamiento y la distribución de los datos y tener una visión cercana de cuáles de los algoritmos de Machine Learning se deben entrenar para conseguir predecir dicho comportamiento con muestras futuras, se visualizó que el conjunto de los datos preprocesado tiene 5000 muestras, 67 características y 1 variable de salida, 2 clases: 0 y 1 y un número total de 2988 muestras para la clase 0 y 2012 muestras para la clase 1. Esto evidencia que no hay un desbalance significativo entre la cantidad de muestras de una clase que, de otra, ya que el problema del desbalance es serio, porque la clase con mayor número de muestras o mayoritaria puede sesgar los resultados y la clase minoritaria se ve mal representada y esto no conduce a realizar correctas predicciones.

También se realizó una exploración gráfica por medio de un gráfico de barras que muestra cómo es la distribución de las muestras por clase como se muestra en la figura 1, y se hizo un gráfico de la matriz de correlación entre variables que mide el grado de dependencia entre las mismas, en donde se observa que el grado de dependencia no es tan alto, tal y como se muestra en la figura 2, lo que permite ahondar más sobre este problema.


*Figura 1. Número de muestras por clase*

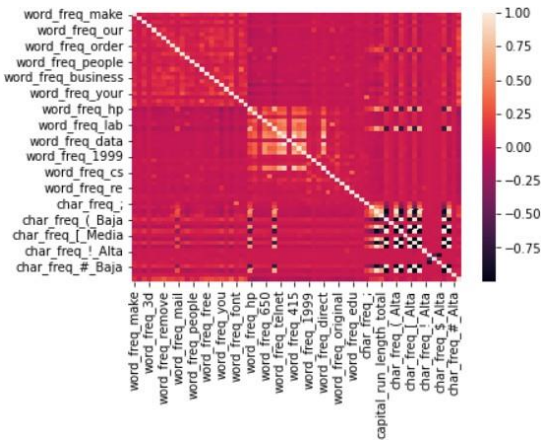



*Figura 2. Matriz de correlación*

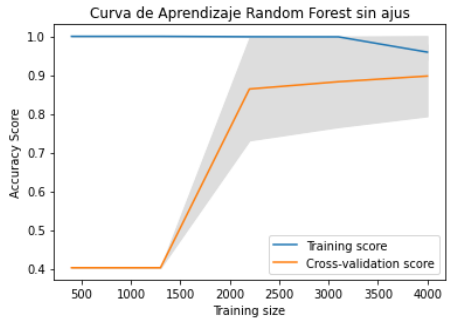
Para este problema de clasificación se considerará la estandarización de los datos o normalización con el fin de que todos ellos tengan la misma escala y los algoritmos de Machine Learning arrojen buenos resultados. Se hizo uso de la estandarización escalada que consiste en que a cada una de las muestras se le resta la media y se divide entre su desviación estándar, tal como se muestra a continuación:

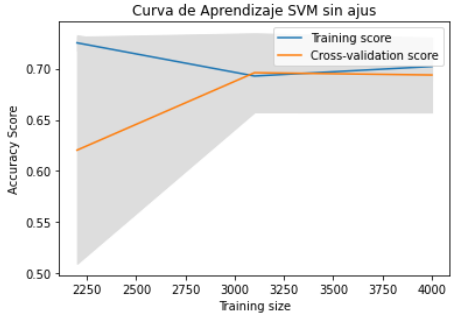
$$X_{normalized} = \frac{X - X_{mean}}{X_{stddev}}$$

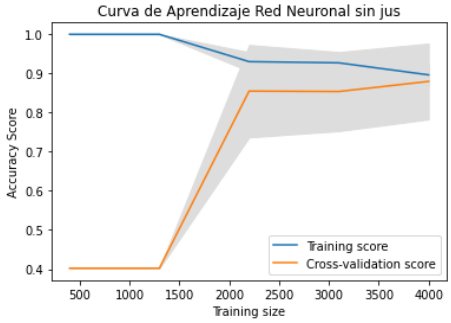
En la matriz de correlación se puede ver gráficamente el grado de dependencia que tiene cada una de las variables respecto a las demás variables. Según la convención establecida por la escala de colores que un color muy oscuro es una correlación baja y que un color muy claro es una correlación muy alta, es decir, 1 indica una correlación muy alta y –1, por el contrario, indica una muy baja. Este diagrama muestra que la correlación en términos generales no es tan alta, porque en muchas regiones se puede observar que el color tiene una tendencia no muy oscura, se puede ver que la correlación está entre –0.25 y 0.5, lo que indica un valor de correlación directa o positiva para el caso de los valores positivos y correlación inversa o negativa para los valores negativos, lo que indica una correlación baja.

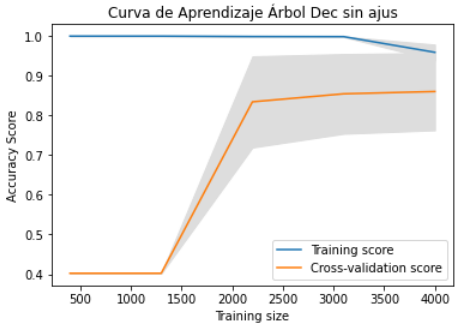
Es útil analizar la relación que tiene la proporción de datos de entrenamiento o el conjunto de entrenamiento con el desempeño, en este caso se mide con el accuracy, debido a que el tamaño del conjunto de entrenamiento puede estar ligado a problemas de sobreajuste o de sub-ajuste, de sobreajuste cuando los modelos se ajustan muy perfectamente a los datos, tanto que dichos algoritmos se ‘aprenden’ los datos y no tienen buena capacidad de generalización o de predicción que es lo que se pretende y de sub-ajuste cuando los modelos no se ajustan bien a los datos y no tienen la capacidad de seguir su comportamiento. También es importante antes de comenzar a entrenar los modelos, saber cuál es la importancia de ajustar los parámetros de los algoritmos, ya que los parámetros mal ajustados pueden desencadenar los problemas antes mencionados: los modelos no siguen la dinámica de los datos garantizando que se llegue al porcentaje de accuracy deseado, que en este caso es del 90%.

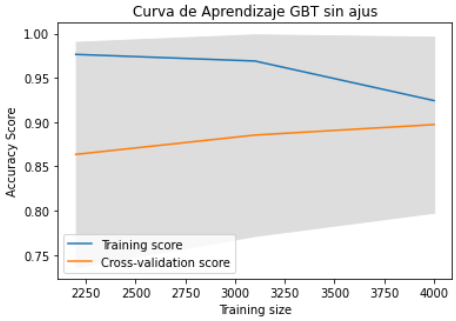
Las curvas de aprendizaje que se trazaron con los modelos cuyos parámetros no están ajustados son:




**Análisis e interpretación de las curvas de aprendizaje de los modelos cuyos parámetros no están ajustados**

Se puede observar que en casi todas las gráficas hay una tendencia al sobreajuste, esto significa que en los modelos que se graficaron, las muestras pueden ser memorizadas por dichos algoritmos, y esto es lo que precisamente se debe evitar, porque lo que se pretende no es que el algoritmo de Machine Learning se aprenda el comportamiento de las muestras muy bien, sino que tenga una buena capacidad de entrenamiento y tenga la capacidad de generalizar o de predecir nuevas muestras que se ingresen. En todas las curvas se evidencia que el error de entrenamiento (training score) o accuracy de entrenamiento es más grande que el error de validación o accuracy de validación, y también en algunas como la del clasificador de red neuronal hay una convergencia de las dos curvas, lo que puede ser beneficioso para el problema.

Por esta razón se deben ajustar los parámetros de los algoritmos que puedan acarrear problema de sobreajuste y subajuste, y así disminuir la complejidad de los mismos, y así como también, como en el caso de la máquina de vectores de soporte hay que ajustar el parámetro de regularización para que el modelo no se sobreajuste, esto es a lo que se le conoce como ajuste de hiper parámetros que se verá en la siguiente sección que es la de Elección del modelo de clasificación.

# 5. Iteraciones de desarrollo

### 3.1 Preprocesado de datos

El conjunto de datos que se utiliza corresponde a un problema de clasificación de tipos de correo en spam y no spam o en ham y no ham, el cual tiene inicialmente 4601 muestras y 57 características y la variable objetivo o de salida, con dos clases que corresponden a 0 y 1, en donde 0 corresponde a los correos que no son spam y 1 a los correos que son spam. Cada uno de estos tipos de correos posee una serie de características de acuerdo a su tipo, a cada correo se le evalúan las mismas características y así mismo tiene un valor distinto en cada una de ellas. Este problema pertenece a un problema supervisado, ya que se conoce de antemano la variable de salida que es la que se quiere predecir.

El conjunto de datos que se pretende obtener tiene las características que son: al menos 5000 muestras, el 10% de las características ha de ser de tipo categórico, el 5% de los datos en al menos 3 columnas deben ser datos faltantes, para ello se crea un dataframe con las muestras faltantes, con la finalidad de que este sea rellenado con datos vacíos o datos faltantes. Algunas de las características del conjunto de datos se codifican como datos o variables categóricas, es decir, en este caso las variables que se simularon para ser las categóricas fueron char_freq_(, char_freq_[, char_freq_!, char_freq_#, y char_freq_$, que se clasificaron en tres categorías, que fueron Baja, Media y Alta dependiendo de la característica de una muestra en específico, para esto se dividió en tres intervalos cada uno de los valores o datos de las muestras de éstas características.

Luego de haber realizado lo anterior, se dispuso a empezar con el preprocesamiento de los datos, el cual corresponde a dejar un conjunto de datos limpio con el fin de que sirva como entrada a los modelos de Machine Learning que se utilizarán posteriormente para resolver el problema de clasificación, esto es, un conjunto de datos sin valores faltantes, con una buena cantidad de muestras y sin variables de tipo categórico, porque los modelos sólo aceptan datos numéricos, cuantitativos de tipo discreto o continuo. El tipo de problema que se va a abordar es de clasificación porque la variable objetivo es de tipo discreta.

Las variables de tipo categórica se codificaron por medio de la codificación One Hot Encoding, esto es que dichas variables quedaron representadas de acuerdo a su categoría como un 1 si la variable está presente y un 0 si la variable está ausente, y para los datos faltantes se realizó una imputación de valores con la función

KNNImputer de la librería de Sklearn de Python, se utilizó esta función, debido a que permite hacer una imputación de valores no aleatorios que tienen como principio de funcionamiento el algoritmo de KNN (k-Nearest-Neighbours), que corresponde a los k vecinos más cercanos del dato en particular.

### 3.2 Modelos supervisados

Antes de empezar a entrenar los diferentes algoritmos de aprendizaje automático con los datos de entrenamiento, es necesario saber cuál será la proporción que se utilizará en el conjunto de entrenamiento y validación o de test y dividir el conjunto de datos en ambos conjuntos mencionados. Se entrenaron los diferentes modelos con la proporción de 0.7 y 0.3 que corresponden a los tamaños de los conjuntos de entrenamiento y de validación respectivamente y se dividieron los datos utilizando la función de *train_test_split* de la librería de Sklearn de Python y seguidamente se procedió a estandarizar los datos como se mencionó anteriormente.

Los siguientes son los modelos que se entrenaron con los conjuntos de entrenamiento y prueba:

#### *3.2.1 Random Forest o Bosque Aleatorio*

Es un tipo de algoritmo de tipo supervisado que pertenece a los métodos de ensamble de árboles de Machine Learning. Lo que hace es que crea un grupo de árboles y el conjunto de variables evaluado en cada nodo se selecciona de forma aleatoria del conjunto de variables originales. En este modelo se evaluaron f1, y matriz de confusión, el error de entrenamiento y prueba, y se pretende ajustar los parámetros que son el número de estimadores y la máxima profundidad que corresponden a los hiper parámetros del modelo. Se probaron con los valores n_estimators = [5,10,20] y max_depth = [7,10,15]. Los resultados de cada una de las iteraciones se muestran como sigue en la *Figura 3*:

| | Árboles | Max_Prof | Error_train | Error_test | F1 | T_N | F_P | F_N | T_P |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 5.0 | 7.0 | 0.887429 | 0.876667 | 0.837291 | 0.559333 | 0.026667 | 0.096667 | 0.317333 |
| **1** | 5.0 | 10.0 | 0.915429 | 0.876667 | 0.838990 | 0.555333 | 0.030667 | 0.092667 | 0.321333 |
| **2** | 5.0 | 15.0 | 0.938286 | 0.894667 | 0.864028 | 0.560000 | 0.026000 | 0.079333 | 0.334667 |
| **3** | 10.0 | 7.0 | 0.898571 | 0.882000 | 0.858287 | 0.524667 | 0.061333 | 0.056667 | 0.357333 |
| **4** | 10.0 | 10.0 | 0.924286 | 0.899333 | 0.870163 | 0.562000 | 0.024000 | 0.076667 | 0.337333 |
| **5** | 10.0 | 15.0 | 0.943714 | 0.896667 | 0.867860 | 0.557333 | 0.028667 | 0.074667 | 0.339333 |
| **6** | 20.0 | 7.0 | 0.902000 | 0.888667 | 0.865431 | 0.530667 | 0.055333 | 0.056000 | 0.358000 |
| **7** | 20.0 | 10.0 | 0.924857 | 0.904000 | 0.876501 | 0.563333 | 0.022667 | 0.073333 | 0.340667 |
| **8** | 20.0 | 15.0 | 0.944571 | 0.909333 | 0.883761 | 0.564667 | 0.021333 | 0.069333 | 0.344667 |

*Figura 3. Resultados de Random Forest**

* La matriz de confusión está representada por T_N, F_P, F_N, T_P para cada una de las iteraciones realizadas

### *3.2.2 Máquina de Vectores de Soporte con Kernel polinomial*

En este modelo se pretende buscar la frontera que minimice el error y que sea la mejor, llamada frontera de decisión. También se evaluaron f1, matriz de confusión, error de entrenamiento y prueba. Los hiper parámetros que se requieren ajustar son los de regularización y de grados del polinomio, ya que el parámetro de regularización controla la cantidad de vectores de soporte del modelo y entre más se regularice el algoritmo, más vectores de soporte hay y entre menos se regularice, menos vectores de soporte existen y más compleja es la frontera de decisión. El grado del polinomio también se debe ajustar porque se aumenta la flexibilidad al tener un polinomio de grado muy alto, ya que se aumenta la complejidad del modelo y así se pueden dibujar fronteras más complejas que representen los datos. Los valores con los que se experimentó fueron C = [0.05,1.5, 9.6], degree = [1,3,4]. Los resultados se muestran como sigue en la *Figura 4:*

| | Reg | Degree | Error_train | Error_test | F1 | T_N | F_P | F_N | T_P |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.05 | 1.0 | 0.840000 | 0.828000 | 0.757062 | 0.560000 | 0.026000 | 0.146000 | 0.268000 |
| **1** | 0.05 | 3.0 | 0.659143 | 0.630000 | 0.199134 | 0.584000 | 0.002000 | 0.368000 | 0.046000 |
| **2** | 0.05 | 4.0 | 0.648000 | 0.620667 | 0.162003 | 0.584000 | 0.002000 | 0.377333 | 0.036667 |
| **3** | 1.50 | 1.0 | 0.884000 | 0.878000 | 0.840453 | 0.556667 | 0.029333 | 0.092667 | 0.321333 |
| **4** | 1.50 | 3.0 | 0.773714 | 0.732667 | 0.539610 | 0.576000 | 0.010000 | 0.257333 | 0.156667 |
| **5** | 1.50 | 4.0 | 0.726571 | 0.674667 | 0.377551 | 0.576000 | 0.010000 | 0.315333 | 0.098667 |
| **6** | 9.60 | 1.0 | 0.889714 | 0.874667 | 0.836806 | 0.553333 | 0.032667 | 0.092667 | 0.321333 |
| **7** | 9.60 | 3.0 | 0.850000 | 0.808000 | 0.715415 | 0.566667 | 0.019333 | 0.172667 | 0.241333 |
| **8** | 9.60 | 4.0 | 0.786286 | 0.728667 | 0.541150 | 0.568667 | 0.017333 | 0.254000 | 0.160000 |

*Figura 4. Resultados SVM kernel polinomial*

### *3.2.3 Máquina de Vectores de Soporte con Kernel RBF*

El algoritmo de SVM con kernel RBF es un algoritmo muy flexible, puesto que puede crear una frontera mucho más compleja que se ajuste a los datos. Al igual que en los algoritmos previos se evaluó el error de entrenamiento y de prueba, f1 y la matriz de confusión. Los valores de los hiper parámetros con los que se experimentó fueron C = [0.05,1.5,9.6] y gamma = [0.1,0.7,4.5] y se obtuvieron los siguientes resultados plasmados en la *Figura 5*:

| | Reg | Gamma | Error_train | Error_test | F1 | T_N | F_P | F_N | T_P |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.05 | 0.1 | 0.735429 | 0.723333 | 0.511190 | 0.578667 | 0.007333 | 0.269333 | 0.144667 |
| **1** | 0.05 | 0.7 | 0.602571 | 0.586000 | 0.000000 | 0.586000 | 0.000000 | 0.414000 | 0.000000 |
| **2** | 0.05 | 4.5 | 0.602571 | 0.586000 | 0.000000 | 0.586000 | 0.000000 | 0.414000 | 0.000000 |
| **3** | 1.50 | 0.1 | 0.932286 | 0.877333 | 0.836590 | 0.563333 | 0.022667 | 0.100000 | 0.314000 |
| **4** | 1.50 | 0.7 | 0.950571 | 0.766667 | 0.611111 | 0.583333 | 0.002667 | 0.230667 | 0.183333 |
| **5** | 1.50 | 4.5 | 0.953429 | 0.729333 | 0.525701 | 0.579333 | 0.006667 | 0.264000 | 0.150000 |
| **6** | 9.60 | 0.1 | 0.946000 | 0.874000 | 0.832891 | 0.560000 | 0.026000 | 0.100000 | 0.314000 |
| **7** | 9.60 | 0.7 | 0.952857 | 0.764000 | 0.611842 | 0.578000 | 0.008000 | 0.228000 | 0.186000 |
| **8** | 9.60 | 4.5 | 0.954286 | 0.729333 | 0.525701 | 0.579333 | 0.006667 | 0.264000 | 0.150000 |

*Figura 5. Resultados SVM kernel RBF*

### *3.2.4 Red Neuronal*

Las redes neuronales son un tipo de algoritmo bioinspirado. En este caso, se entrenó un MLP (Multi Layer Percepron) o Perceptrón Multicapa, que tiene al perceptrón como la unidad básica de las redes neuronales. Se experimentó con una capa oculta y con distintas cantidades de neuronas, con la función de activación 'tangh' o tangente hiperbólica, con el parámetro max_iter de 100 y con el parámetro random_state = 1. No se entrenó al modelo con más capas ocultas, porque a medida que crece el número de estas capas, aumenta el costo computacional y el algoritmo no tiende a mejorar. Las neuronas fueron 5,7,9. Se evaluó los errores de entrenamiento y prueba, f1 y la matriz de confusión. Los resultados se muestran como sigue en la *Figura 6:*

| | Capas Ocultas | Neuronas | Error Ent | Error test | F1 | T_N | F_P | F_N | T_I |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 5.0 | 0.902857 | 0.888000 | 0.857868 | 0.550000 | 0.036000 | 0.076000 | 0.338000 |
| **1** | 1.0 | 7.0 | 0.903429 | 0.893333 | 0.876352 | 0.515333 | 0.070667 | 0.036000 | 0.378000 |
| **2** | 1.0 | 9.0 | 0.905143 | 0.890667 | 0.861252 | 0.551333 | 0.034667 | 0.074667 | 0.339333 |

*Figura 6. Resultados de la red neuronal MLP*

### *3.2.5 Decisión Tree o Árbol de Decisión*

El algoritmo de árbol de decisión subdivide las muestras en el espacio de características hasta lograr grupos con comportamientos similares a partir de los cuales se pueden hacer predicciones. Existen unos parámetros que se deben ajustar para que el modelo realice buenas predicciones y tenga un buen comportamiento con los datos de entrenamiento. La profundidad se debe controlar, al igual que el número de características en el árbol. Se experimentó con los árboles de decisión con los parámetros max_depth = [3,4,8]y max_features = [5,9,10], al igual que se midió los errores, f1, matriz de confusión. Los resultados de las iteraciones con estos parámetros se muestran en la *Figura 7.*

| | Max_Prof | Max_Car | Error_train | Error_test | F1 | T_N | F_P | F_N | T_P |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3.0 | 5.0 | 0.775429 | 0.769333 | 0.741405 | 0.438667 | 0.147333 | 0.083333 | 0.330667 |
| **1** | 3.0 | 9.0 | 0.804571 | 0.802000 | 0.698477 | 0.572667 | 0.013333 | 0.184667 | 0.229333 |
| **2** | 3.0 | 10.0 | 0.797143 | 0.794667 | 0.696252 | 0.559333 | 0.026667 | 0.178667 | 0.235333 |
| **3** | 4.0 | 5.0 | 0.824571 | 0.814000 | 0.790383 | 0.463333 | 0.122667 | 0.063333 | 0.350667 |
| **4** | 4.0 | 9.0 | 0.809143 | 0.806667 | 0.712871 | 0.566667 | 0.019333 | 0.174000 | 0.240000 |
| **5** | 4.0 | 10.0 | 0.806286 | 0.806000 | 0.723647 | 0.552000 | 0.034000 | 0.160000 | 0.254000 |
| **6** | 8.0 | 5.0 | 0.869143 | 0.849333 | 0.797853 | 0.552000 | 0.034000 | 0.116667 | 0.297333 |
| **7** | 8.0 | 9.0 | 0.878000 | 0.870667 | 0.845787 | 0.516000 | 0.070000 | 0.059333 | 0.354667 |
| **8** | 8.0 | 10.0 | 0.872000 | 0.845333 | 0.822630 | 0.486667 | 0.099333 | 0.055333 | 0.358667 |

*Figura 7. Resultados Árbol de decisión*

### *3.2.6 Gradient Boosting Tree*

El método de Gradient Boosting Tree es un método de tipo Boosting que entrena modelos secuencialmente con el fin de que las muestras que no han podido ser modeladas correctamente se modelen en los siguientes modelos. El árbol de decisión es el clasificador base en este tipo de modelos. Se ajustan los parámetros de n_estimators y max_depth que pueden generar sobreajuste. Los valores con los que se entrenó el modelo fueron n_estimators = [3,5,7] y max_depth = [5,6,8]. Los resultados obtenidos en cada una de las iteraciones están en la *Figura 8.*

| | Estiimadores | Max_Prof | Err_train | Err_test | F1 | T_N | F_P | F_N | T_P |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3.0 | 5.0 | 0.846286 | 0.827333 | 0.746327 | 0.573333 | 0.012667 | 0.160000 | 0.254000 |
| **1** | 3.0 | 6.0 | 0.870286 | 0.848667 | 0.785241 | 0.572000 | 0.014000 | 0.137333 | 0.276667 |
| **2** | 3.0 | 8.0 | 0.910857 | 0.886667 | 0.849558 | 0.566667 | 0.019333 | 0.094000 | 0.320000 |
| **3** | 5.0 | 5.0 | 0.880286 | 0.863333 | 0.813127 | 0.566000 | 0.020000 | 0.116667 | 0.297333 |
| **4** | 5.0 | 6.0 | 0.891143 | 0.872000 | 0.826715 | 0.566667 | 0.019333 | 0.108667 | 0.305333 |
| **5** | 5.0 | 8.0 | 0.921429 | 0.887333 | 0.852402 | 0.562000 | 0.024000 | 0.088667 | 0.325333 |
| **6** | 7.0 | 5.0 | 0.887429 | 0.873333 | 0.829749 | 0.564667 | 0.021333 | 0.105333 | 0.308667 |
| **7** | 7.0 | 6.0 | 0.900857 | 0.883333 | 0.844996 | 0.565333 | 0.020667 | 0.096000 | 0.318000 |
| **8** | 7.0 | 8.0 | 0.922000 | 0.893333 | 0.862543 | 0.558667 | 0.027333 | 0.079333 | 0.334667 |

*Figura 8. Resultados GBT*

## 3.4 Resultados, métricas y curvas de aprendizaje

Como se observó en la sección anterior de los modelos supervisados, se mostraron los resultados obtenidos con cada una de las iteraciones de los entrenamientos de los diferentes modelos. Ahora se va a elegir entre todos los posibles modelos el mejor modelo para cada tipo de modelo con los mejores hiper parámetros, para esto se utiliza la función *Grid Search* de la librería de Python.

Aplicando la anterior función a cada uno de los modelos los resultados fueron:

| Modelo | Mejores parámetros | Accuracy |
|---|---|---|
| Random Forest | n_estimators=20, max_depth=15 | 0.8988 |
| SVM kernel polinomial | C=9.6, degree=1 | 0.8814 |
| SVM kernel RBF | C=9.6, gamma=0.1 | 0.8642 |
| Red Neuronal | hidden_layer_size=(5,) | 0.8897 |
| Decision Tree | max_depth=8, max_features=9 | 0.8491 |
| Gradient Boosting Tree | max_depth=8, n_estimators=7 | 0.8808 |

*Figura 10. Mejores resultados con mejor métrica*

A continuación, se muestra un diagrama de barras con los porcentajes de accuracy para cada uno de los modelos entrenados con los mejores parámetros seleccionados, en la *Figura 11*.

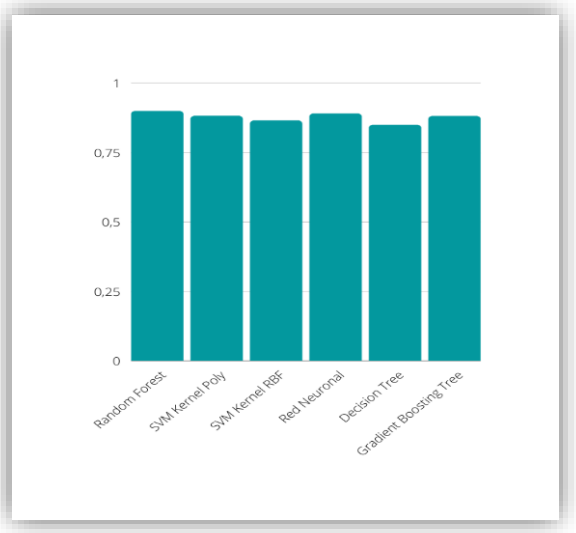



*Figura 11. Diagrama de los mejores accuracy*

De la figura anterior se puede ver que el mejor modelo que se entrenó con el mejor accuracy fue el de Random Forest, ya que se obtuvo un accuracy de 0.8988, que se puede aproximar a 0.9 lo que equivale al 90% del rendimiento correcto o del rendimiento deseado del modelo.

### 3.3 Modelo no supervisado

Una vez realizado el análisis con solamente métodos supervisados, se realiza un análisis implementando métodos no supervisados a modo de tratamiento de los datos y optimización de la cantidad de variables utilizada. En este caso se utilizó el análisis por componentes principales (PCA), de esta manera se podrá reducir la dimensionalidad del dataset y conservar aquellos datos que conserven la mayor cantidad de información.

### *3.3.1 Análisis por componentes principales (PCA):*

El PCA permite reducir la dimensionalidad de los datos de una muestra, mientras que permite también la conservación de la información, de esta forma es posible analizar y determinar la cantidad de datos necesaria para obtener un porcentaje de evaluación satisfactoria, como se muestra en la figura 9, una gráfica que relaciona el porcentaje de varianza acumulada con el número de componentes principales, para obtener un accuracy cercano al 90% tal y como se define en la métrica de evaluación es necesario tener al menos 45 componentes principales del dataset de estudio.

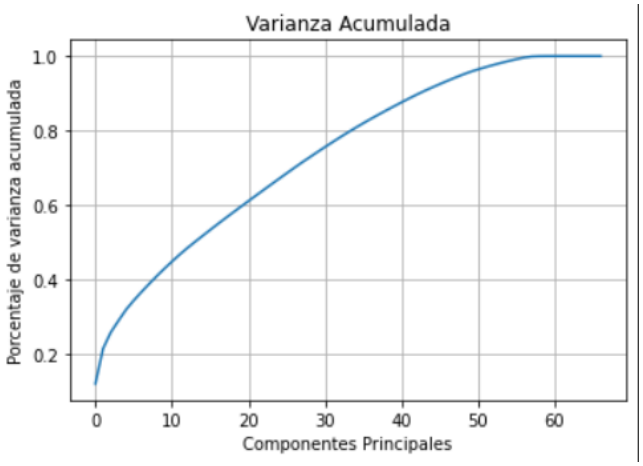



Figura 9. Gráfica de varianza acumulada

Para dar claridad a lo dicho en el anterior párrafo, se evaluó el modelo supervisado con mejor desempeño (Random Forest), con una reducción significativa de variables y presentando el accuracy obtenido en cada una, tal y como se muestra en la tabla 1, en donde se evidencia que el accuracy obtenido es cercano al 90% aunque el número de variables se vea reducido.

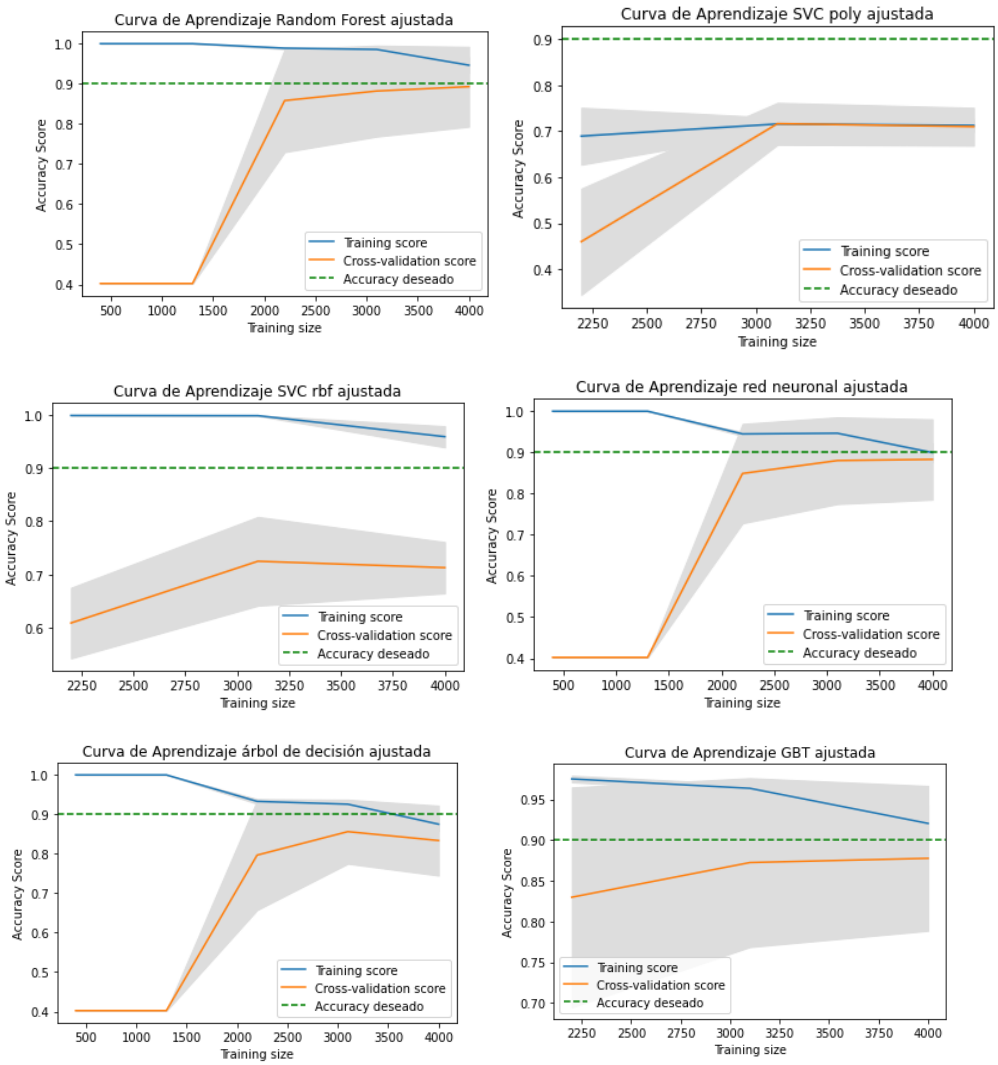
*Tabla 1. Resultados obtenidos con una reducción de variables empleando el modelo de Random Forest*

| Número de variables | Reducción [%] | Accuracy |
|---|---|---|
| 10 | 14.92 | 0.88 |
| 20 | 29.85 | 0.89 |
| 30 | 44.78 | 0.89 |
| 55 | 82.09 | 0.89 |

## Curvas de aprendizaje de los mejores modelos

Las curvas de aprendizaje son un buen elemento que permite visualizar gráficamente el desempeño o rendimiento de los modelos de acuerdo a una métrica. Se realizaron

las curvas de aprendizaje de los modelos entrenados con los mejores hiper parámetros las cuales se muestran a continuación.




**Interpretación de las curvas de aprendizaje de los mejores modelos**

Teniendo en cuenta la métrica que se evalúa en el conjunto de datos que es el Accuracy y que se desea tener un valor del mismo de 0.90 (90%) en la exactitud que los modelos arrojaron para las diferentes iteraciones hechas anteriormente, de las curvas de aprendizaje se puede analizar lo siguiente:

- Random Forest: La curva de aprendiza tiene una tendencia a tener un buen equilibrio entre el sesgo (bias) y la varianza (variance), ya que se puede observar que ambas curvas, de entrenamiento y validación tienen la tendencia hacia el accuracy deseado. Si se aumenta la cantidad de muestras o instancias el accuracy de validación tiende a aumentar y el de entrenamiento tiende a disminuir, esto es debido a que a medida que aumenta el número de muestras el modelo pierde capacidad de entrenamiento y gana capacidad de generalización ante muestras nuevas.

- Máquina de vectores de soporte con kernel polinomial: En la gráfica se puede ver que ambas curvas están lejanas de la métrica que se desea, esto significa que existe un alto sesgo en los datos, lo que quiere decir que puede haber una inclinación de los datos hacia una clase u otra dependiendo del número de muestras en cada una. Puede haber problemas de sobreajuste.
- Máquina de vectores de soporte con kernel rbf: Se observa que la curva de entrenamiento tiene la tendencia de ir decreciendo y que la curva de validación va decreciendo muy lentamente, lo que significa que el modelo va perdiendo capacidad para generalizar o predecir muestras nuevas y así como también a medida que va cayendo la curva de entramiento respecto a la cantidad de muestras en el conjunto de entrenamiento, cae la capacidad de entrenar las muestras. Este modelo presenta una alta varianza en los datos.
- Red Neuronal: En la gráfica se puede observar que este modelo presenta una alta tendencia a tener un buen balance entre el sesgo y la varianza, porque ambas curvas se acercan cada vez más a el valor deseado de la métrica de evaluación. A medida que se hace más grande la cantidad de muestras la capacidad de generalización del modelo tiene un muy leve decrecimiento y se puede decir que puede llegar un punto en que se mantenga constante y la capacidad del entrenamiento de los datos va disminuyendo. Es un buen modelo, hay un buen acercamiento al valor deseado.
- Árbol de Decisión: Este modelo presenta una curva de aprendizaje en la cual se puede observar que la curva de entrenamiento sobrepasa el valor de accuracy deseado, lo que es beneficioso y también significa que a medida que se va incrementando la cantidad de muestras en el conjunto de entrenamiento el modelo va perdiendo capacidad de entrenamiento; con respecto a la curva de validación se puede ver que no está tan cerca del rendimiento deseado y que tiene una leve tendencia a decrecer. Podría llegar a ser un buen modelo. Acá se puede ver el comportamiento de la alta varianza en los datos, la curva de validación se va alejando del valor de desempeño deseado.
- Gradient Boosting Tree: En esta gráfica se puede apreciar que ambas curvas, la de entrenamiento y de test se acercan a la línea de desempeño deseado, lo que puede ser beneficioso en cierta medida, aunque la curva de validación tiene una tendencia a decrecer de manera leve a medida que se aumenta la cantidad de muestras en el conjunto de entrenamiento del modelo. En este diagrama puede haber alta varianza en los datos, ya que ambas curvas presentan cierta lejanía al accuracy deseado.

# 6. Retos y consideraciones de despliegue

Luego de realizar todo el proceso que se requiere para un proyecto de Machine Learning o de Aprendizaje Automático, en la última etapa que es la de despliegue es adecuado considerar que partiendo de los resultados que se obtuvieron cuando se entrenaron los modelos con los mejores hiper parámetros, se debe tener en cuenta que es necesario realizar experimentos constantemente con el fin de que el sistema se encuentre actualizado y funcione de la mejor manera posible, es decir, realice predicciones correctas en un tiempo razonable.

Como reto se tiene el establecer un mejor desempeño en las métricas de evaluación, buscando que el modelo sea cada vez más eficaz en el análisis de los correos no deseados.

# 7. Conclusiones

- Es necesario explorar antes de entrenar los modelos con el conjunto de datos, dado que en muchas ocasiones y en este caso la cantidad de muestras o de registros no es suficiente para obtener buenas predicciones y no representan la tendencia de los datos.
- Las curvas de aprendizaje son una herramienta que es muy útil cuando se desea saber sobre el desempeño o rendimiento de los modelos, esto permite conocer cuáles son los errores de entrenamiento y de validación.
- El ajuste de los parámetros que pueden generar los fenómenos de sobreajuste y subajuste es muy relevante para que el desempeño de los modelos sea óptimo.
- En general los mejores modelos entrenados tuvieron buen desempeño, todos estuvieron muy cerca del 90% deseado.
- Los métodos de aprendizaje no supervisado permiten extraer las características más importantes de los datos de evaluación y también a reducir sus dimensiones, conservando la información más importante.
- Un número de variables grande no siempre determina que la información que suministran es de calidad o que presenta relevancia a la hora de hacer una análisis, ya que como se vio en el estudio hecho con el PCA no fue necesario utilizar la totalidad de muestras para obtener un buen desempeño.