Default credit app: Sistema para detectar incumplimiento en créditos bancarios utilizando técnicas de máchine learning

Daniel Felipe Rivera Arroyave
Facultad de Ingeniería.
Universidad de Antioquia.
Medellín, Colombia
Daniel.riveraa@udea.edu.co.

***Resumen*—La detección adecuada de clientes que puedan incumplir en el pago de créditos bancarios es un problema presente en todas las entidades bancarias y que se ha maneja a través de los años con diversos métodos de solución. En este trabajo se presenta una solución basada en modelos de máchine learning como lo son: Análisis discriminante cuadrático, Ventana de Parzen, Random Forest, Redes Neuronales Artificiales y Máquinas de soporte Vectorial, utilizando un conjunto de datos tomado del año 2005 en una entidad bancaria, se utilizaron diversas técnicas propias de las ciencias de datos con el fin de optimizar las predicciones, todos los resultados fueron documentados y los mejores modelos fueron desplegados en un aplicativo web.**

***Index Terms*—Machine learning, Default credit card, Cross validation, Modelos predictivos, Accuracy-score.**

I. Introducción

El *machine learning* como ciencia de datos de datos no es algo nuevo y sus fundamentos teóricos se remontan a los años 50 pero no es sino hasta los últimos años años donde se ha experimentado un boom tecnológico a nivel de hardware que ha posibilitado explorarlo como solución en multitud de problemas de diferente índole, las entidades bancarias que por su modelo de negocio deben estar innovando en soluciones o alternativas a sus problemas no son ajenas al uso de este tipo de técnicas [1]. En este documento se pretende solucionar un problema real al que pueden estar sometidas las entidades bancarias donde se parte de un conjunto de características personales y un historial crediticio de los clientes con el fin de determinar si esta persona realizará el pago de sus obligaciones crediticias el mes siguiente o no, la base de datos corresponde a una entidad bancaria real en Taiwan del año 2005 [2], se utilizaron diferentes modelos clásicos del *machine learning* sobre los que se hizo un proceso general definido para solucionar problemas estándar con el fin de maximizar su resultado bajo unas métricas de validación.

En el resto del artículo se presenta la información detallada del problema y explicación de la base de datos en la sección II, una revisión del estado del arte y trabajos de referencia en la sección IV, los experimentos preliminares con los modelos se encuentran en la sección V, el análisis de las características en la sección VI, mientras que la selección y extracción características son comentadas en las secciones VII y VIII respectivamente, finalmente las conclusiones y comparación de los resultados frente al estado del arte se trabaja en la sección IX.

II. Comprensión del problema

El problema abordado es un problema de *machine learning* de tipo supervisado el cual corresponde a una clasificación binaria en la que pretende determinar si una persona con ciertas características va cumplir o incumplir con el pago de la deuda de su tarjeta de crédito en el próximo mes, el campo de aplicación principal son entidades bancarias donde funcionaria como un sistema de apoyo al experto.

III. Descripción de la base de datos

La base de datos se llama *"Default of credit card clients dataset"*, se encuentra alojada por la UCI desde el 2016 y contiene información crediticia de personas en Taiwan durante el periodo de abril y septiembre de 2005, la base de datos cuenta con 23 características y 30.000 registros sin datos faltantes pero con algunos no documentados, una última característica corresponde a si la persona realizará el pago del próximo mes o no y es esta característica la que es usada como variable respuesta. A continuación se presentan todas las variables con su explicación e interpretación.

III-A. Variables de entrada

- LIMIT_BAL: Monto del crédito de la persona - Continuo.
- SEX: Género de la persona.
  - 1: Masculino.
  - 2: Femenino.
- EDUCATION: Nivel educativo de la persona.
  - 1: Postgrado.
  - 2: Título universitario.
  - 3: Bachillerato.
  - 4: Otros.
- MARRIAGE: Estado civil de la persona.
  - 1: Casad@.
  - 2: Solter@.
  - 3: Otros.
- AGE: Edad en años de la persona -Continuo/Discreto.
- PAY_1,2,3,4,5: Variables correspondientes a 5 meses que indican el estado de los pagos durante los meses comprendidos entre abril y septiembre de 2005.
  - -1: Pagar debidamente.
  - 1-9: Cantidad de meses de retraso en el pago.

- BILL_AMT1,2,3,4,5,6: Variables correspondientes al monto de dinero disponible en la cuenta los meses anteriores comprendidos entre abril y septiembre de 2005 en dólares-Continuo.
- PAY_AMT1,2,3,4,5,6: Variables correspondientes al monto de dinero pagado o abonado al crédito en los meses anteriores comprendidos entre abril y septiembre de 2005 en dólares-Continuo.

III-B. Variable de salida

- DEF_PAY: Indica se se realizará o no el pago del crédito en el próximo mes.
  - 0: No.
  - 1: Sí.

IV. Trabajos relacionados

Con el fin de obtener un punto de referencia e información de utilidad en el proyecto, se presenta de forma resumida una revisión bibliográfica de 4 proyectos que utilizaron el mismo conjunto de datos para resolver este problema.

IV-A. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients [3]

En este artículo no mencionan nada respecto al gran desbalance que hay en los datos, la técnica de validación fue de tipo boostraping pero no especifican si fue estratificada o no, ni la partición de los datos, sin embargo una de las métricas de validación fue el *ROC AUC* que es conveniente para este tipo de problemas con datos des-balanceados. Los modelos que evaluaron fueron: *Artificial neural networks, classification trees, naive Bayesian classifier, K-nearest neighbor classifiers, logistic regression y discriminant analysis* lo cual es conveniente para comparar con los resultados de este proyecto en la sección IX, respecto a los resultados que presentan demuestran que el modelo *Artificial neural network* fue el que obtuvo mayor desempeño con un valor de 0.54 en *ROC AUC* y una tasa de error de 0.17, en cuanto a la métrica de *ROC AUC* los otros modelos presentaron resultados similares mientras que hay una diferencia notable en el *Accuracy* entre este y los demás modelos.

IV-B. Predicting Credit Card Defaults with Deep Learning and other Machine Learning Models [4]

Este artículo es mucho más interesante que el anterior debido a que aplica técnicas de *deep learning* y compara con modelos de *machine learning* tradicionales para evaluar el desempeño, los modelos utilizados son *KNN, Logistic Regression ,Naive Bayes, Decision tree C4.5 y DNN* siendo esta último modelo el que presentó mejores resultados con una reducción de características, no mencionan la metodología de validación, respecto a las métricas de desempeño se midió el *Accuracy*, *Precision* y *Recall*. Realizan un análisis muy particular con el árbol de decisión al elegir el tamaño máximo de partición con base en k-means que es una técnica de clustering, la selección de características que realizaron fue con el índice Grey. También advierten que debido a la naturaleza del problema donde los bancos pierden mucho dinero por el incumplimiento de los clientes en sus pagos el *Recall* es una muy buena medida de desempeño por considerar los falsos negativos.

IV-C. An experimental comparison of classification algorithms for imbalanced credit scoring data sets [5]

En este artículo se dedican a comparar varias técnicas que se pueden utilizar en el análisis de conjuntos de datos con información crediticia que se encuentren desequilibrados debido a que es un problema común en este contexto *Regresión logística,Análisis discriminante cuadrático,Redes neuronales,SVM, Arbol de decisión, Knn, Random forest, Gradiente boost* que es una gran variedad de modelos y los resultados pueden compararse fácilmente en la sección IX sin embargo utilizan 5 conjuntos de datos diferentes, por lo que es de esperarse no hacen un análisis muy profundo, la metodología de validación que utilizaron fue de tipo bootstraping sin especificar el número de iteraciones proponen como métrica de validación el área bajo la curva, estadística de friedman y pruebas post hoc de Nemenyi. Los modelos con mejores resultados fueron el random forest y el gradient boost, son los primeros en mencionar técnicas de sobremuestreo y submuestreo, siendo el primero el que dió mejores resultados.

IV-D. Application of Machine Learning Algorithms in Credit Card Default Payment Prediction [6]

Este artículo sólo utiliza modelos de *machine learning* tradicionales, combinados con técnicas de ensamble, los modelos utilizados fueron Logistic Regression, Decision tree, SVM, Naive Bayes Knn, Bagging, stacking y Boosting que fueron evaluados bajo una metodología de validación cruzada de 10 folds, no mencionan ninguna consideración sobre el desbalance del conjunto de datos, mientras que sus métricas de validación fueron *Accuracy* donde todos los modelos superaron el valor de 0.82 , *Recall* y *especificidad*, además de hicieron también un extracción de características utilizando como criterio el *método Wrapper* sin mencionar el tipo de selección o las características resultantes. Los modelos con mejores resultados fueron *SVM* y *Logistic Regression*. Algo destacable de este artículo es la eliminación valores atípicos y extremos logrando así una mayor limpieza de los datos.

V. Experimentos

En el repositorio donde se adquirió la base de datos no reporta valores faltantes, sin embargo en algunas variables categóricas se encuentran etiquetas que la documentación no menciona y se trataron con base a la siguiente interpretación:

- EDUCATION: Etiquetas 0, 5 y 6 no están documentadas, un total de 345 muestras son asignados a la clase 4 :Otros.
- MARRIAGE: Etiqueta 0 no está documentada, 54 muestras son asignadas a la clase 3: Otros.
- PAY_N°: Etiquetas -2 y 0 no se encuentran documentadas. Siguiendo la lógica implementada para estas etiquetas donde -1 significa que pagó a tiempo y los números positivos son el número de meses de retraso en el pago, se

puede inferir que las etiquetas -2 y cero también significa que se pagó a tiempo y no hay retrasos. Por lo tanto se usará la categoría 0 para agrupar a las etiquetas -2 y -1, y en consecuencia su significado cambiará a pagar a tiempo. esta decisión se toma considerando que la etiqueta 0 agrupa mayor cantidad de datos que las otras en cuestión, esta es una decisión crítica sobre que puede impactar sobre los modelos.

La distribución de muestras por clase en la variable de salida es la siguiente 0=No:23364 1=Sí:6636, cuya magnitud puede ser apreciada en la figura 1.

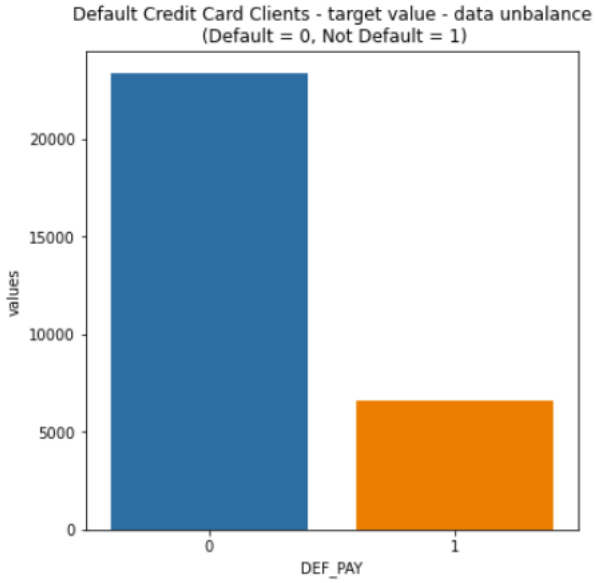

Figura 1: Distribución de muestras por clase en la variable respuesta.

Como se puede apreciar en la figura 1, la variable respuesta se encuentra altamente des-balanceada y esto fue tenido en consideración en su respectivo momento de los experimentos usando una metodología de validación cruzada estratificada de 10 folds además de seguir las recomendaciones mencionada en la revisión bibliográfica de la sección IV sobre el uso de técnicas de *Oversampling* o *Undersampling* con el objetivo de mejorar los resultados, adicional a esto como es común en problemas de *machine learning* fue necesario aplicar otra técnica de *One hot encoding* [7] sobre las características categóricas que no presentan ninguna clase de orden, estas características son *SEX*, *EDUCATION, MARRIAGE*, finalmente fue necesario llevar las distintas variables a una misma escala, el tipo de escalamiento es de tipo estándar, la figura 2 usada para representar las características estandarizadas fue tomada de un repositorio en *Kaggle [8]* donde abordaron el mismo problema.

En el desarrollo se presentó una etapa importante llamada experimentos donde se implementaron distintas instancias de modelos de *machine learning*, concretamente *Análisis discriminante Cuadrático, Ventana de Parzen, Random Forest, Perceptron Multicapa y Máquinas de soporte Vectorial* que

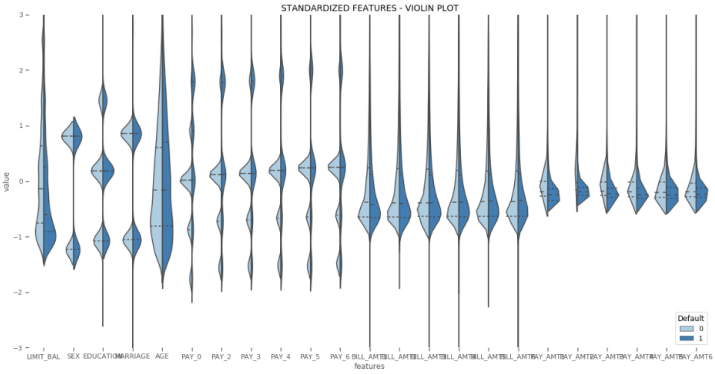

Figura 2: Gráfico de violín de características estandarizadas

fueron entrenados y validados de forma estratificada como se mencionó anteriormente, las métricas de validación utilizada fueron el área bajo la curva roc *ROC AUC*, *Balanced Accuracy* y el *D1-score* que son complementadas con su respectivo intervalo de confianza para las dos primeras que son a las que se les dio mayor prioridad, es importante aclarar que todos los modelos utilizados fueron tomados de la librería de *Scikit-learn* [9] y que para la búsqueda de hiper-parámetros se utilizó el total de muestras, todos los resultados serán presentados a continuación junto con un su respectiva matriz de confusión y gráfica de curva ROC.

V-A. Análisis discriminante cuadrático.

Este fue el primer modelo y más simple en ejecutarse, destacó por su corto tiempo de ejecución en comparación con los que serán abordados próximamente, el hiper-parámetro que se varió fue el *reg_param* debido a su corto tiempo de ejecución fue posible probar numerosas permutaciones con estos hiper-parámetros, sin embargo no se obtuvieron muy buenos resultados y la regularización disponible no logró contrarrestar el des-balance de los datos, los mejores resultados se obtuvieron con un parámetro de regularización de 0.7 y se presentan en la tabla I, la 3 son los resultados gráficos sin aplicar técnicas de SMOTE mientras que en la figura 4 si fueron usadas, en la segunda matriz de confusión se presenta es notable una mejora respecto a la primera en la capacidad de clasificar las muestras de clase 1, a cambio de clasificar un poco peor las muestras de clase 0.

Cuadro I: Resultados obtenidos análisis discriminante cuadrático

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
No	0.750552	0.022238	0.686759	0.020097	0.70
Sí	0.747692	0.022371	0.693352	0.021174	0.66

V-B. Ventana de Parzen

Este modelo fue el segundo más simple en ejecutarse pero se caracteriza debido a que aunque se usó la librería *scikit-learn* [9], esta no provee una instancia como tal con el algoritmo de Ventana de Parzen y fue necesario utilizar un estimador

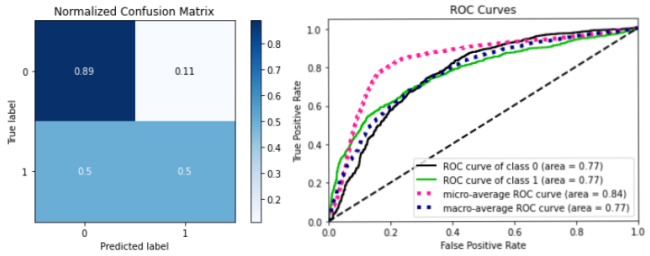

Figura 3: Resultados gráficos análisis discriminante cuadrático sin SMOTE

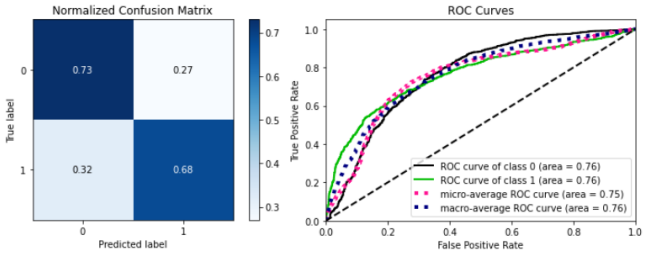

Figura 4: Resultados gráficos análisis discriminante cuadrático con SMOTE

base de *Kernel Density* por cada clase para lograr implementar el algoritmo, el tipo de kernel utilizado siempre fue de tipo *gaussiano* donde el único hiper-parámetro que se varió fue el ancho de la ventana entre 0.05 y 1, al igual que el modelo anterior no obtuvo muy buenos resultados y fue fuertemente afectado por el des-balance debido al funcionamiento interno del algoritmo, de aquí se puede hacer un análisis preliminar sobre los datos que refleja que las clases no se encuentran muy separadas entre ellas, los mejores resultados que se obtuvieron fue con un ancho de ventana de 0.95, el uso de SMOTE no reflejó mayor cambio positivo en los resultados de los experimentos debido a que en las métricas de validación el *ROC AUC* se redujo en 0.004 y el *Balanced Accuracy* aumentó en 0.05 aproximadamente, esto se puede puede evidenciar en la tabla II.

Cuadro II: Resultados Ventana de Parzen

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
No	0.719846	0.009918	0.616317	0.000486	0.63
Sí	0.715065	0.009239	0.667799	0.011768	0.66

V-C. Random Forest

El primer modelo complejo utilizado fue *Random Forest* que es una modificación del algoritmo de árboles de decisión para clasificación, la cual consiste no sólo en crear un grupo de árboles B, sino también en incluir un componente aleatorio en la partición que se realiza en cada nodo. El componente aleatorio incluido en Random Forest es la elección del conjunto de variables a evaluar en cada nodo y la decisión final se toma a partir de de la combinación de decisiones de los B árboles, los hiper-parámetros que se variaron fueron el número de estimadores, la cantidad máxima de características

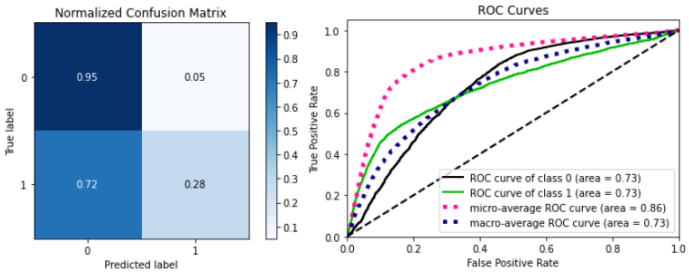

Figura 5: Resultados gráficos análisis Ventana de Parzen sin SMOTE.

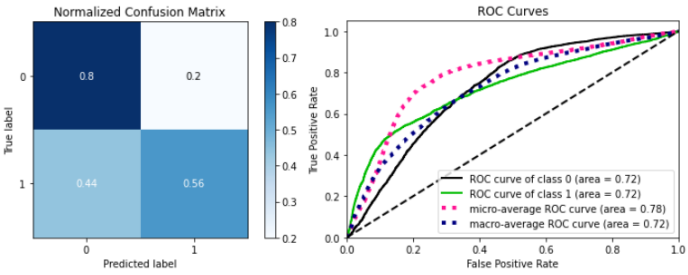

Figura 6: Resultados gráficos análisis Ventana de Parzen con SMOTE.

por nodo y el criterio de división, los resultados presentan una mejora sustancial respecto a los dos anteriores y pueden deberse en parte al parámetro de **class_weight** que se estableció para todos en ”balanced”para que el modelo tuviera en consideración la cantidad de muestras por clase. Los mejores hiper-parámetros fueron *criterion='entropy'*, *max_features=4*, *n_estimators=250*, este parámetro de *class_weight* no genera ningún efecto al momento de aplicar SMOTE por lo que puede ser la justificación de la poca diferencia en los resultados respecto a la aplicación o no de la técnica.

Cuadro III: Resultados Random Forest

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
No	0.769956	0.01883	0.651536	0.011217	0.68
Sí	0.759301	0.017279	0.689405	0.013982	0.70

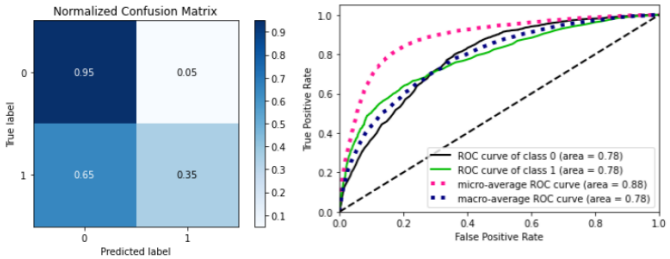

Figura 7: Matriz de confusión Random Forest sin SMOTE

V-D. Redes Neuronales Artificiales

Se utilizo una red neuronal de tipo *Perceptron Multicapa* compuesta por un conjunto de unidades más pequeñas

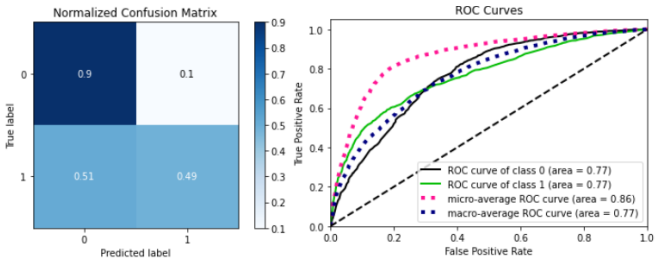

Figura 8: Curva roc Random Forest sin SMOTE

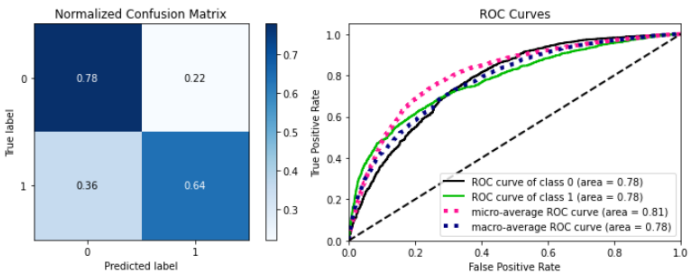

Figura 10: Resultados gráficos análisis MLP con SMOTE

llamados perceptrones simples. Esta red neuronal tiene la capacidad de procesar cálculos más complejos a medida que se le agregan más perceptrones haciéndolo un algoritmo potente, este modelo fue el que mejores resultados presentó de todos los que se evaluaron, sin embargo la diferencia respecto al Random Forest y Máquina de Soporte Vectorial es mínima además de estar ubicado en medio de estos dos respecto al tiempo de ejecución, los hiper-parámetros que se variaron fueron el tipo de activación en la capa oculta, la tasa de aprendizaje y el número de neuronas en la capa oculta para encontrar que los mejores valores fueron *activation='tanh'*, *alpha=0.001*, *hidden_layer_sizes=70)* además de utilizar una parada anticipada con el fin de evitar el sobre entrenamiento, por otra parte el uso de SMOTE reflejo un cambio positivo significativo en el *Balanced Accuracy* pero disminuyo levemente el *ROC AUC*, los resultados obtenidos se presentan en la tabla V y figuras 9 y 10.

Cuadro IV: Resultados RNA

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
No	0.772875	0.017632	0.658641	0.017816	0.68
Sí	0.767977	0.017584	0.701556	0.01566	0.68

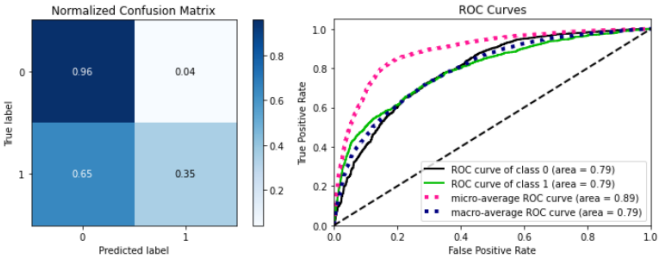

Figura 9: Resultados gráficos análisis MLP sin SMOTE

V-E. Máquinas de soporte vectorial

Para encontrar la mejor solución, este modelo utiliza como criterio la maximización del margen, el cual se entiende como la distancia más corta entre la frontera de decisión y cualquiera de las muestras. Para este modelo, se crean vectores de soporte y un vector de soporte es aquel que se encuentra en el punto más cercano a la frontera, por otra parte es el modelo más costoso computacionalmente respecto a los anteriores y se ve reflejado en sus tiempos de ejecución, al igual que el random forest presenta un hiper-parámetro de *class_weight* para indicarle que debe de tener consideración la cantidad de muestras por clase al momento de ajustarse, los hiper-parámetros que se variaron fueron el parámetro de penalización C, el valor de gamma y el tipo de kernel entre lineal o rbf. Los mejores resultados se obtuvieron con *C=0.1* y un *gamma=0.01*, también es importante mencionar que para poder evaluar las métricas de validación es necesario contar con el parámetro de *probaility=True*. Respecto a los resultados fueron similares al Random Forest y al Perceptron Multicapa, donde el uso de SMOTE no reflejó cambios significativos en el *ROC AUC* pero aumentó en 0.05 el *Balanced Accuracy*.

Cuadro V: Resultados SVM

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
No	0.766029	0.017768	0.659808	0.010845	0.70
Sí	0.765609	0.017416	0.7018	0.012478	0.70

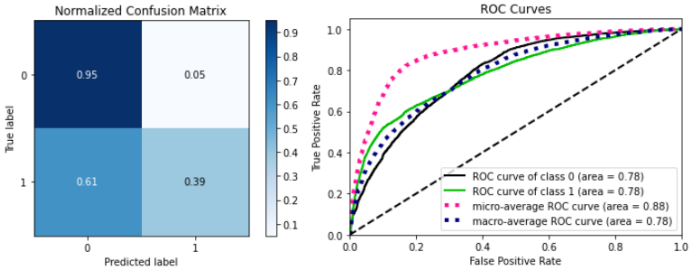

Figura 11: Resultados gráficos análisis SVM sin SMOTE

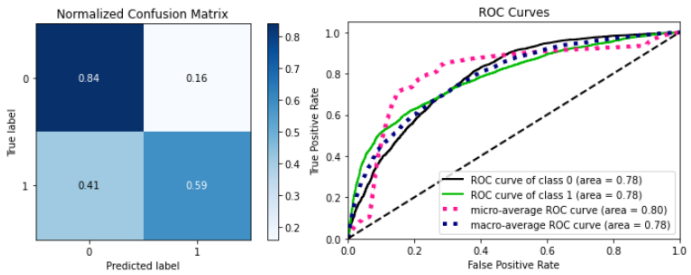

Figura 12: Resultados gráficos análisis SVM con SMOTE

Con base en los resultados de los experimentos se concluyó que los mejores modelos para trabajar con el conjunto de datos en orden de puntuación en *ROC AUC* son *RNA Perceptron Multicapa*, *Máquina de soporte vectorial* y *Random Forest* y es

con estos con los se trabajó en las siguientes secciones, además de que el uso de SMOTE representa mejorías sobre el *balanced accuracy* pero no en el *ROC AUC*, un experimento similar en donde se instanció un modelo de tipo MLP y entrenó con un sub-conjunto de datos perfectamente balanceado demostró que la dificultad de los modelos para generalizarse con estos datos no se debe solamente al des-balance porque los resultados fueron similares a los 3 anteriores obteniendo un valor de 0.764084 en *ROC AUC* y un *Balanced ccuracy* de 0.698088.

VI. Análisis discriminante

Para problemas de clasificación es fundamental que las características sean altamente discriminantes, sin embargo es prudente incluir en esta sección un análisis de correlación entre las características para identificar una posible información redundante, la idea de esto y el fragmente de código necesario fue tomado del repositorio de Kaggle [8] mencionado en la sección V, un mapa de calor de esto es la figura 13 donde se aprecia que las características mas fuertemente correlacionadas con las las de PAY_N° y BILL_AMTN° lo que es entendible teniendo en cuenta que estas representan meses acumulativos.

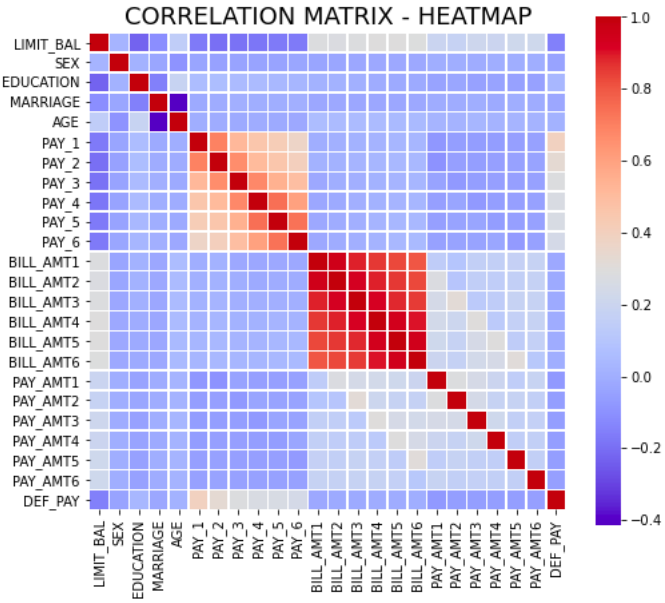

Figura 13: Correlación entre características.

Por otra parte, el análisis discriminante de fisher realizado a cada característica, arrojó que las que tiene mayor poder discriminante son SEX y MARRIAGE siendo las únicas en superar el valor de 0.5 como se ve en la figura 14, sugiriendo que estas sean las únicas a tener en cuenta por los modelos y explicando por qué en la sección V no se obtuvieron muy buenos resultados.

VII. Selección de características

Para la selección de características se usó una selección secuencial hacia adelante pero en su variación flotante, Este método parte desde una característica inicial y agrega nuevas características con el fin de incrementar el valor en el

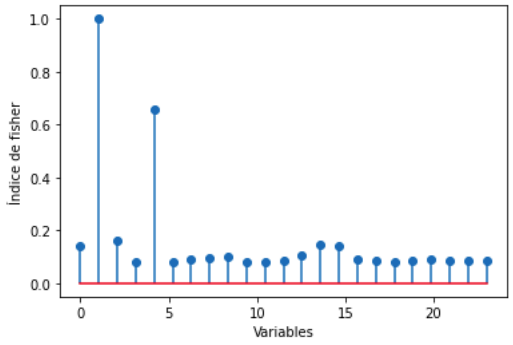

Figura 14: Análisis discriminante de fisher

criterio de selección, el criterio de selección fue de tipo *Wrapper* utilizando las mismas métricas de validación que los experimentos en la sección V *ROC AUC* justificado por el des-balance de los datos, mientras que el único estimador base fue un modelo de tipo *Perceptron Multicapa* debido a que este fue el que obtuvo mayor puntuación con esta métrica, agregando a que el costo computacional de hacer esta selección de características con una *Máquina de Soporte de Vectores* es muy alto y que los hiper parámetros del *Random Forest* se ajustan con base a las características por lo que los hiper parámetros seleccionados inicialmente posiblemente no arrojen la mejor combinación de características, los resultados obtenidos arrojan que con 21 características se obtiene mejores resultados, en este caso las características a descartar son PAY_4, PAY_AMT5 y PAY_AMT6 sin embargo la diferencia sigue siendo mínima respecto a los resultados obtenidos con las 23 originales, 18, 22, 19 y 16. Finalmente se decidió que estas 16 características son la mejor opción de la selección considerando los resultados respecto a las originales y que son aproximadamente un 30 % menos de características, en contraste con lo analizado en la sección VI se nota el poco poder discriminante entre las características y que en las 16 seleccionadas se encuentran las 2 que sobrepasaron el valor de 0.5 en el índice de fisher, sin embargo estas sólo son seleccionadas después de 14 características.

Los resultados de esto se evaluaron nuevamente con los 3 modelos seleccionados con sus mejores hiper-parametros y aplicando SMOTE, dichos resultados se resumen en la tabla VI y las figuras 15, 16 y 17.

Cuadro VI: Resumen de resultados con características seleccionadas

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
MLP	0.773508	0.020914	0.704647	0.01221	0.70
SVM	0.769759	0.019155	0.701504	0.0119	0.69
RF	0.748127	0.017248	0.683909	0.014916	0.70

VIII. Extracción de características

Una alternativa a la selección de características es la extracción y una de las más utilizadas es *PCA [10]* por sus siglas en inglés de análisis de componentes principales, con

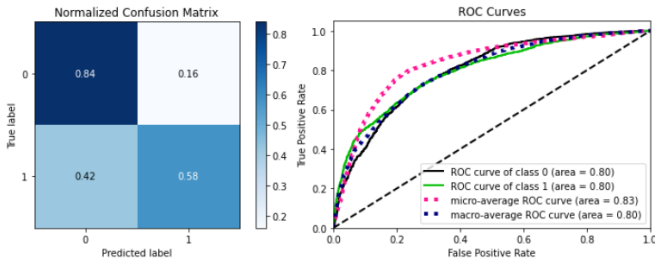

Figura 15: Resultados gráficos Perceptron Multicapa con selección de características

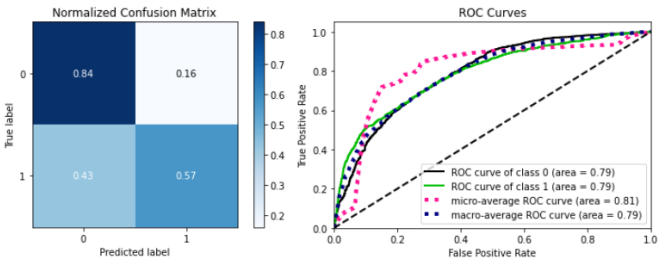

Figura 16: Resultados gráficos SVM con selección de características

esta técnica el objetivo es reducir las características iniciales a N variables nuevas o componentes, siendo N un número menor a la cantidad de variables originales, esta transformación se realiza buscando un número pequeño de N pero conservando la mayor cantidad de información posible, esta información está asociada al nivel de variación dentro de las variables, una forma de estimar un número óptimo de componentes es realizando una gráfica de N componentes vs Varianza acumulada y elegir, esta gráfica se realiza en la figura 18 donde se puede ver que con 18 componentes se tiene representado más del 95 % de la información de los datos y puede ser un número óptimo de componentes, sin embargo sigue siendo necesario validar esta elección de componentes principales con un estimador. Nuevamente se utilizó el *Perceptron Multicapa* como estimador base para evaluar el desempeño y complementar la decisión del mejor número de componentes, los resultados indican que 21 componentes son las que mejor desempeño presentan después del conjunto original con un *Balanced accuracy* de 0.655 y un *ROC AUC* 0.761 de , pero

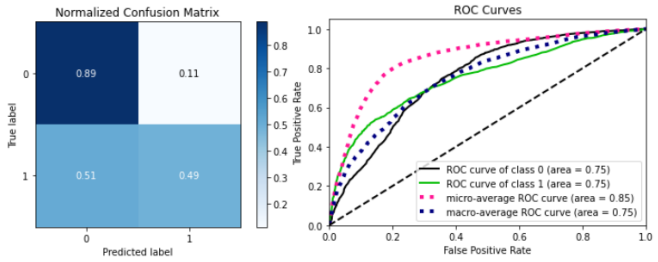

Figura 17: Resultados gráficos Random Forest con selección de características

sus resultados son muy similares a 17 componentes con 0.768 y 0.654 respectivamente, por lo que se eligió este número ya que implica una mayor reducción de variables. Los resultados finales en las métricas de validación después de evaluar los tres mejores modelos utilizando este número se ven en la tabla VII mientras que los resultados gráficos se ven en las figures 19, 20 y 21

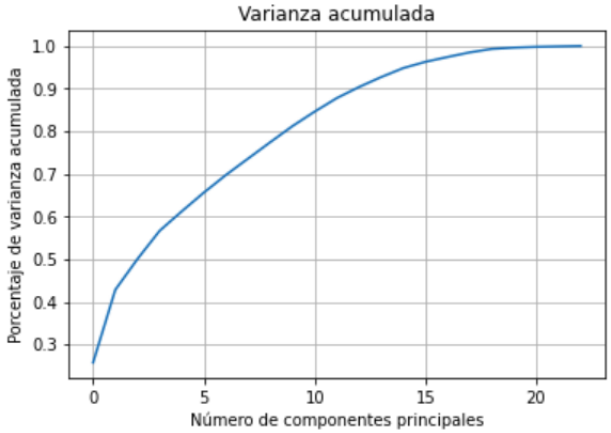

Figura 18: N componente vs Varianza acumulada

Cuadro VII: Resumen de resultados con características extraidas

SMOTE	AUC	AUC IC	B_Acc	B_Acc IC	F1-score
RNA	0.768055	0.021075	0.700981	0.015506	0.78
SVM	0.769759	0.019155	0.701504	0.0119	0.69
RF	0.749229	0.017574	0.680985	0.014078	0.69

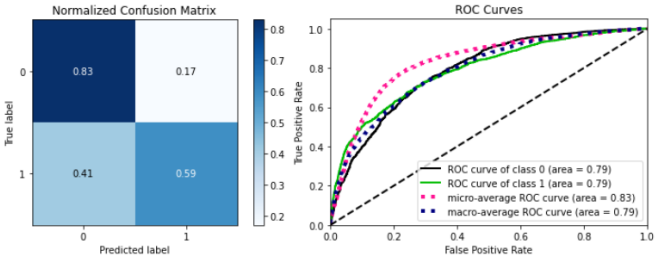

Figura 19: Resultados gráficos Perceptron Multicapa con extracción de características

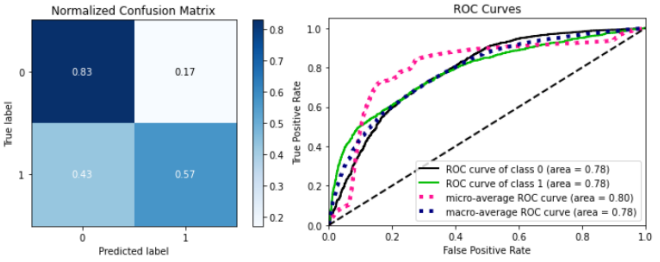

Figura 20: Resultados gráficos SVM con extracción de características

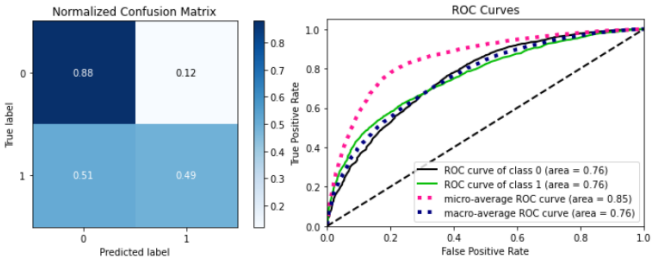

Figura 21: Resultados gráficos Random Forest con extracción de características

IX. Discusión de resultados y conclusiones

Con respecto a los resultados obtenidos en este proyecto y los documentados en la sección IV, especificamente [3], [5] y [6] se concuerda que los modelos de *Redes Neuronales Artificiales* y *Máquinas de soporte Vectorial*, sólo es posible compararlos objetivamente bajo la métrica de validación del *ROC AUC*, ya que en la revisión bibliográfica no se mencionaban si el *Accuracy* era balanceado o no, mientras que en este trabajo siempre se utilizó el tipo balanceado, además de metodologías de validación de tipo estratificada para asegurar que los resultados no fueran sesgados por las particiones.

El uso de técnicas de sobremuestreo como SMOTE en los experimentos de la sección V presentó mejoras aproximadas del 5 % respecto al *Balanced Accuracy* y empeoró en aproximadamente un 1 % el *ROC AUC* lo cual concuerda parcialmente con los resultados obtenidos en el artículo [5]. Si bien la primera no es la métrica de validación principal durante este proyecto, se tuvo en consideración su mejoría y se utilizó esta técnica para las validaciones finales respecto a la selección y extracción de las características en las secciones VII y VIII.

En el artículo [3] el mejor modelo fue *RNA* con un valor de *ROC AUC* de 0.54, además del *Análisis discriminante cuadrático* que obtuvo 0.43 bajo esta misma métrica, estos valores no son concordantes respecto este proyecto donde el valor es superado por más de 0.2, en lo que si concuerdan es que la diferencia entre los resultados de los distintos modelos es muy poca respecto al *Accuracy* sin importar que este es balanceado o no.

En la selección de características ocurrió un fenómeno muy interesante y es que las características más discriminantes no aparecieron desde el principio en la selección secuencial y no fue si no hasta después de 13 características que aparecen, esto contradice un análisis preliminar de la sección VI donde podría esperarse que estas dos características al ser las más discriminantes según el índice de Fisher fueran las primeras en aparecer y que el resto de características se fueran agregando al resultante de la selección secuencial en orden de mayor a menor con base a esta medida de Fisher, por otra lado la mejor combinación de características no logró representar mayor cambio sobre los experimentos que utilizaban el conjunto total y no se pudo concluir verdaderamente que alguna de las características aportara ruido y optimizar el resultado, sin embargo se llegó a la conclusión de que el mejor modelo fue el *RNA* aplicando selección de características con base a que es mucho más liviano por su dimensionalidad frente al conjunto original y a la extracción presentando resultados muy similares.

La extracción de características fue igual de interesante porque se esperaba una disminución en el valor de los resultados considerando la naturaleza de la misma donde se emplearon un 70 % de componentes principales originales que terminan por representar menor cantidad de varianza acumulada, o lo que es igual a menos información y lo que verdaderamente se obtuvo fue nuevamente resultados muy similares respecto al conjunto original y el resultante de la selección.

Algo a destacar es que estos resultados obtenidos son altamente dependientes del contexto al que pertenecen los datos: **Créditos bancarios en una entidad bancaria específica de Taiwan durante el año 2005** y no existe garantía respecto a los resultados en un contexto diferente aunque se empleen características similares, en dicho caso debería repetirse todo el proceso presentado en este trabajo y se esperaría que los resultados fuesen diferentes.

Para terminar, el objetivo de este trabajo era aplicar un ciclo de desarrollo normal dentro de un problema de máchine learning con el objetivo de determinar cuál era el mejor estimador y las características o variables base, para ello se inició buscando referentes bibliográficos donde resolvieran el mismo problema, luego se realizó una búsqueda de hiper-parámetros en diferentes modelos que finalmente fueron evaluados bajo unas métricas que consideraban la naturaleza de los datos, posterior a eso se realizó un análisis, selección y extracción de características que nuevamente fueron evaluadas pero esta vez con los 3 mejores modelos, al final no fue posible tomar una decisión con certeza y se optó por implementar un aplicativo web donde el usuario carga los datos y es su elección decidir a cuál estimador hacerle caso [1]..

Referencias

[1] C. Chakraborty and A. Joseph, "Machine Learning at Central Banks," Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 3031796, Sep. 2017. [Online]. Available: https://papers.ssrn.com/abstract=3031796

[2] "UCI Machine Learning Repository: default of credit card clients Data Set." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients

[3] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, Part 1, pp. 2473–2480, Mar. 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417407006719

[4] T. Chou and M. Lo, "Predicting credit card defaults with deep learning and other machine learning models," *International Journal of Computer Theory and Engineering*, vol. 10, pp. 105–110, 01 2018.

[5] "An experimental comparison of classification algorithms for imbalanced credit scoring data sets," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3446–3453, Feb. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741101342X

[1] El aplicativo web se encuentra desplegado y puede ser consultado e usado en https://sht97.github.io/credit-card-front/ mientras que el código fuente de este proyecto puede serconsultado en https://github.com/Sht97/Default-of-credit-card-clients-project

[6] "Application Of Machine Learning Algorithms In Credit Card Default Payment Prediction, IJSR - International Journal of Scientific Research(IJSR), IJSR | World Wide Journals." [Online]. Available: https://www.worldwidejournals.com/international-journal-of-scientific-research-(IJSR)/article/application-of-machine-learning-algorithms-in-credit-card-default-payment-p MTcwMzU=/?is=1&b1=57&k=15

[7] M. Cassel and F. Lima, "Evaluating one-hot encoding finite state machines for SEU reliability in SRAM-based FPGAs," in *12th IEEE International On-Line Testing Symposium (IOLTS'06)*, Jul. 2006, pp. 6 pp.–, iSSN: 1942-9401.

[8] "Prediction of Credit Card Default." [Online]. Available: https://kaggle.com/selener/prediction-of-credit-card-default

[9] "User guide: contents — scikit-learn 0.24.1 documentation." [Online]. Available: https://scikit-learn.org/stable/user_guide.html

[10] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101

APÉNDICE

A. *Búsqueda de hiper-parámetros*

	params	mean_test_AUC	std_test_AUC	mean_test_Accuracy	std_test_Accuracy
43	{'reg_param': 0.7, 'tol': 0.1}	0.750336	0.011050	0.797625	0.006471
42	{'reg_param': 0.7, 'tol': 0.01}	0.750336	0.011050	0.797625	0.006471
41	{'reg_param': 0.7, 'tol': 0.001}	0.750336	0.011050	0.797625	0.006471
40	{'reg_param': 0.7, 'tol': 0.0001}	0.750336	0.011050	0.797625	0.006471
39	{'reg_param': 0.5, 'tol': 0.1}	0.747391	0.010679	0.789250	0.007056
38	{'reg_param': 0.5, 'tol': 0.01}	0.747391	0.010679	0.789250	0.007056
37	{'reg_param': 0.5, 'tol': 0.001}	0.747391	0.010679	0.789250	0.007056
36	{'reg_param': 0.5, 'tol': 0.0001}	0.747391	0.010679	0.789250	0.007056
33	{'reg_param': 0.4, 'tol': 0.001}	0.746102	0.010615	0.773250	0.007190
32	{'reg_param': 0.4, 'tol': 0.0001}	0.746102	0.010615	0.773250	0.007190
34	{'reg_param': 0.4, 'tol': 0.01}	0.746102	0.010615	0.773250	0.007190
35	{'reg_param': 0.4, 'tol': 0.1}	0.746102	0.010615	0.773250	0.007190
31	{'reg_param': 0.3, 'tol': 0.1}	0.745047	0.010944	0.761750	0.007486
30	{'reg_param': 0.3, 'tol': 0.01}	0.745047	0.010944	0.761750	0.007486
29	{'reg_param': 0.3, 'tol': 0.001}	0.745047	0.010944	0.761750	0.007486
28	{'reg_param': 0.3, 'tol': 0.0001}	0.745047	0.010944	0.761750	0.007486
27	{'reg_param': 0.2, 'tol': 0.1}	0.744048	0.011114	0.752375	0.008258
26	{'reg_param': 0.2, 'tol': 0.01}	0.744048	0.011114	0.752375	0.008258
25	{'reg_param': 0.2, 'tol': 0.001}	0.744048	0.011114	0.752375	0.008258
24	{'reg_param': 0.2, 'tol': 0.0001}	0.744048	0.011114	0.752375	0.008258

Figura 22: Análisis discriminante cuadrático

	BandWith	AUC medio	AUC intervalo de confianza	Accuracy medio	Accuracy intervalo de confianza
0	0.05	0.537081	0.012372	0.533374	0.005154
1	0.15	0.575003	0.012653	0.572527	0.002854
2	0.25	0.614641	0.000849	0.602717	0.001761
3	0.35	0.634007	0.005536	0.612028	0.002724
4	0.45	0.652178	0.001392	0.619193	0.002296
5	0.55	0.674952	0.006718	0.624127	0.004692
6	0.65	0.687269	0.007481	0.625064	0.004528
7	0.75	0.697174	0.007663	0.624702	0.003816
8	0.85	0.708546	0.010051	0.621754	0.004112
9	0.95	0.719078	0.010326	0.616694	0.000128

Figura 23: Ventana de parzen

B. *Extracción de características*

	params	mean_test_AUC	std_test_AUC	mean_test_Accuracy	std_test_Accuracy
34	{'criterion': 'entropy', 'max_features': 4, 'n...	0.770496	0.003907	0.649984	0.004424
5	{'criterion': 'gini', 'max_features': 4, 'n_es...	0.769598	0.004801	0.649230	0.002711
55	{'criterion': 'entropy', 'max_features': 'auto...	0.769028	0.005829	0.652867	0.003792
54	{'criterion': 'entropy', 'max_features': 'auto...	0.768878	0.005667	0.654104	0.004651
27	{'criterion': 'gini', 'max_features': 'auto', ...	0.768798	0.004693	0.649676	0.005133
33	{'criterion': 'entropy', 'max_features': 4, 'n...	0.768634	0.005753	0.651784	0.001827
6	{'criterion': 'gini', 'max_features': 4, 'n_es...	0.768210	0.004363	0.646641	0.004579
26	{'criterion': 'gini', 'max_features': 'auto', ...	0.767867	0.006683	0.649912	0.003237
40	{'criterion': 'entropy', 'max_features': 8, 'n...	0.767066	0.005406	0.656907	0.004075
53	{'criterion': 'entropy', 'max_features': 'auto...	0.766709	0.005620	0.652280	0.003207
41	{'criterion': 'entropy', 'max_features': 8, 'n...	0.766567	0.006048	0.655929	0.003916
48	{'criterion': 'entropy', 'max_features': 12, '...	0.766094	0.007107	0.654771	0.005203
32	{'criterion': 'entropy', 'max_features': 4, 'n...	0.765970	0.006393	0.652764	0.002687
47	{'criterion': 'entropy', 'max_features': 12, '...	0.765877	0.005183	0.655784	0.003694
12	{'criterion': 'gini', 'max_features': 8, 'n_es...	0.765816	0.006846	0.650487	0.003396
4	{'criterion': 'gini', 'max_features': 4, 'n_es...	0.765142	0.002862	0.646739	0.001465
13	{'criterion': 'gini', 'max_features': 8, 'n_es...	0.765077	0.005352	0.650994	0.002707
20	{'criterion': 'gini', 'max_features': 12, 'n_e...	0.763497	0.005452	0.651200	0.004017
31	{'criterion': 'entropy', 'max_features': 4, 'n...	0.763355	0.007433	0.647803	0.004327
25	{'criterion': 'gini', 'max_features': 'auto', ...	0.762932	0.006849	0.645487	0.004481

Figura 24: Random Forest

	params	mean_test_AUC	std_test_AUC	mean_test_Accuracy	std_test_Accuracy
47	{'activation': 'tanh', 'alpha': 0.001, 'hidden...	0.775941	0.000435	0.665266	0.002405
40	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.775756	0.000735	0.663162	0.001486
32	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.774783	0.000875	0.666210	0.003262
37	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.774519	0.000503	0.649834	0.006606
30	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.773558	0.000763	0.659701	0.002219
31	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.773028	0.000145	0.663222	0.003561
48	{'activation': 'tanh', 'alpha': 0.001, 'hidden...	0.772143	0.001360	0.664506	0.001485
34	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.771797	0.000239	0.662137	0.003048
51	{'activation': 'tanh', 'alpha': 0.001, 'hidden...	0.771738	0.002921	0.656182	0.000646
20	{'activation': 'logistic', 'alpha': 0.001, 'hi...	0.771559	0.000138	0.660351	0.006606
42	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.771381	0.003397	0.664478	0.003582
45	{'activation': 'tanh', 'alpha': 0.001, 'hidden...	0.771308	0.001025	0.664044	0.002488
41	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.771037	0.001642	0.662066	0.005708
29	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.770494	0.000792	0.661372	0.000015
27	{'activation': 'tanh', 'alpha': 0.1, 'hidden_l...	0.770388	0.000966	0.656821	0.002007
36	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.770380	0.003824	0.658837	0.001038
44	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	0.769770	0.006172	0.658037	0.004407

Figura 25: Perceptron Multicapa

	params	mean_test_AUC	std_test_AUC	mean_test_Accuracy	std_test_Accuracy
18	{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}	0.767202	0.002979	0.695905	0.003001
26	{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}	0.765331	0.004086	0.702122	0.004448
16	{'C': 0.1, 'gamma': 0.001, 'kernel': 'rbf'}	0.765285	0.000732	0.686918	0.003499
32	{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}	0.764906	0.004240	0.700475	0.005317
24	{'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}	0.764305	0.004710	0.688200	0.003924
10	{'C': 0.01, 'gamma': 0.01, 'kernel': 'rbf'}	0.760817	0.001036	0.682175	0.002814
34	{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.758294	0.005265	0.701536	0.005667
8	{'C': 0.01, 'gamma': 0.001, 'kernel': 'rbf'}	0.757046	0.001021	0.590236	0.006007
0	{'C': 0.001, 'gamma': 0.001, 'kernel': 'rbf'}	0.756672	0.000578	0.500000	0.000000
9	{'C': 0.01, 'gamma': 0.001, 'kernel': 'linear'}	0.755109	0.003777	0.691154	0.007865
11	{'C': 0.01, 'gamma': 0.01, 'kernel': 'linear'}	0.755109	0.003777	0.691154	0.007865
13	{'C': 0.01, 'gamma': 0.1, 'kernel': 'linear'}	0.755109	0.003777	0.691154	0.007865
15	{'C': 0.01, 'gamma': 1, 'kernel': 'linear'}	0.755109	0.003777	0.691154	0.007865
7	{'C': 0.001, 'gamma': 1, 'kernel': 'linear'}	0.755109	0.003819	0.685174	0.004065
1	{'C': 0.001, 'gamma': 0.001, 'kernel': 'linear'}	0.755109	0.003819	0.685174	0.004065
5	{'C': 0.001, 'gamma': 0.1, 'kernel': 'linear'}	0.755109	0.003819	0.685174	0.004065
3	{'C': 0.001, 'gamma': 0.01, 'kernel': 'linear'}	0.755109	0.003819	0.685174	0.004065

Figura 26: Máquina de soporte vectorial

	CON_EXTRA	NUM_COMP	BALANCE_ACC	IC_ACC	ROC_AUC	IC_ROC_AUC	T_EJECUCION
7	SI	21.0	0.656321	0.013621	0.771004	0.020334	3.177561
4	SI	17.0	0.656724	0.012211	0.770380	0.017859	3.177561
6	SI	20.0	0.657952	0.013611	0.770235	0.017526	3.177561
5	SI	18.0	0.658930	0.011356	0.769424	0.018842	3.177561
0	NO	23.0	0.658357	0.021383	0.768352	0.021616	3.177561
3	SI	15.0	0.647444	0.013269	0.767130	0.022190	3.177561
2	SI	13.0	0.639666	0.014428	0.764036	0.020800	3.177561
1	SI	12.0	0.640437	0.011842	0.762138	0.018057	3.177561

Figura 27: Análisis de componentes principales

C. *Selección de características*

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
21	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14...	[0.7498765395486053, 0.7506231584834298, 0.779...	0.774003	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14...	0.0261464	0.0203428	0.0101714
18	(0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 1...	[0.7518212028335947, 0.7490152910682687, 0.784...	0.773357	(0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 1...	0.0245172	0.0190753	0.00953763
22	(0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	[0.7532687202147632, 0.7465618439137369, 0.783...	0.772821	(0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	0.024422	0.0190011	0.00950055
19	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15...	[0.7498908035824805, 0.7452288483715152, 0.781...	0.772436	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15...	0.027142	0.0211174	0.0105587
16	(0, 1, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, ...	[0.7491207875165043, 0.7500527731419298, 0.783...	0.772254	(0, 1, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, ...	0.0242119	0.0188377	0.00941885
13	(0, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, 18, 20)	[0.7548346210183199, 0.7449682482267983, 0.785...	0.771995	(0, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, 18, 20)	0.0236567	0.0184057	0.00920284
20	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15...	[0.7514442938706882, 0.7513435179181145, 0.777...	0.771867	(0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 15...	0.0223778	0.0174107	0.00870533
14	(0, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, 18, 19...	[0.7557101264647631, 0.7436339625848503, 0.781...	0.771593	(0, 5, 6, 7, 8, 10, 11, 12, 13, 15, 17, 18, 19...	0.0237737	0.0184967	0.00924837

Figura 28: Selección de características