Análisis del conjunto de datos FICO y técnicas de aprendizaje automático

Informe del proyecto final de Aprendizaje Automático

IMAT-ICAI 2024

Miguel Ángel Vallejo de Bergia

Índice:

1. Introducción
2. Análisis exploratorio de los datos
3. Algoritmos de Clasificación
4. Aprendizaje no supervisado
5. Preguntas interesantes e hipótesis
6. Bibliografía

1. Introducción

Para el proyecto final de esta asignatura, he analizado el conjunto de datos FICO, que contiene información sobre las líneas de crédito HELOC. El objetivo principal es determinar si un cliente pagará su préstamo HELOC en un periodo de 2 años. El dataset contiene varias variables como ExternalRiskEstimate o NumTotalTrades. La variable a predecir, que depende de todas las demás es RiskPerformance, donde 0 indica que paga en un periodo de 2 años y 1 indica lo contrario.

Primero se ha hecho un análisis exploratorio de los datos. Luego se han implementado diversos algoritmos de aprendizaje automático para este problema de clasificación. Además, se han utilizado técnicas de aprendizaje no supervisado, con el objetivo de reducir la dimensionalidad del conjunto de datos y de agrupar los datos en grupos o clústeres.

A continuación se analizarán los resultados obtenidos.

2. Análisis exploratorio de los datos

En esta primera parte, se ha realizado un análisis minucioso de los datos, proporcionados por el profesor, y se han preparado los datos para entrenar los modelos de clasificación.

En primer lugar, se hizo una limpieza de datos. Esta consistió en eliminar valores nulos, eliminar valores poco interpretables, eliminar datos atípicos y normalizar los datos.

Después se hizo un estudio de las distribuciones de las variables del dataset y se hicieron contrastes no paramétricos (visuales y no visuales) para estudiar la normalidad de los datos, de los que se obtuvo que ninguna variable del dataset podía aproximarse por una normal.

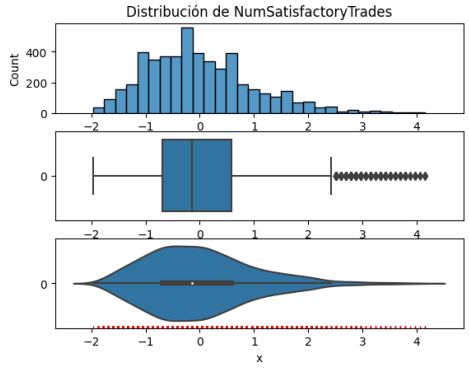

Fig. 1 Distribucion NumSatisfactoryTrades

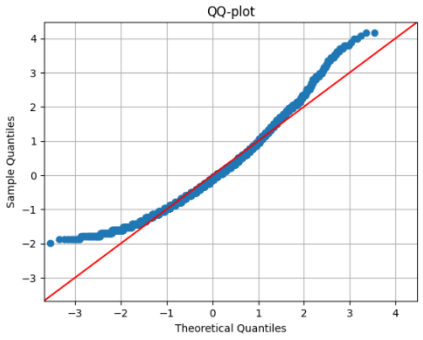

Fig. 2 QQPlot NumSatisfactoryTrades

También se hizo un estudio de las correlaciones y del pps score, que es una medida de la importancia de una cierta variable en una variable de salida.

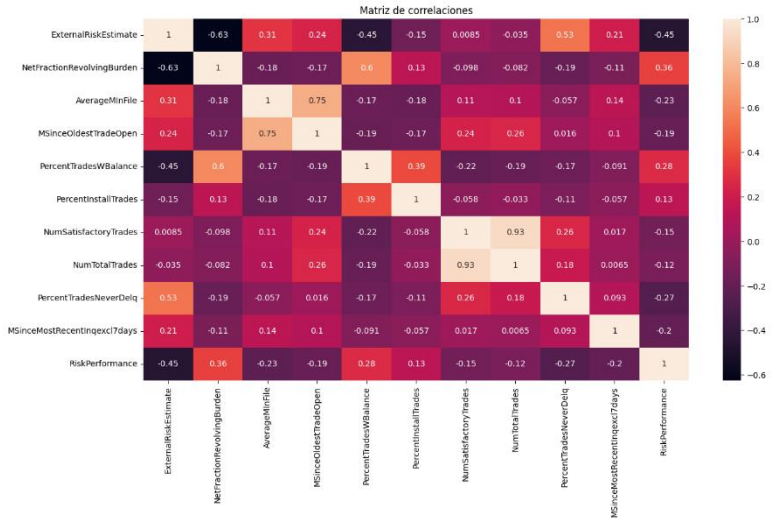

Fig. 3 Matriz de correlaciones

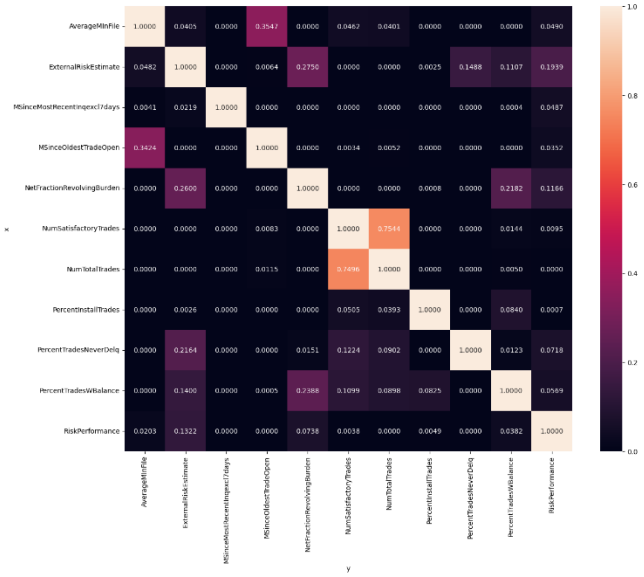

Fig. 4 Matriz pps score

Se observa que las variables más correlacionadas con la variable objetivo son las que más pps score tienen y por ende las más significativas a la hora de hacer la clasificación.

Por último, se procedió a crear los conjuntos train y test, para los modelos de clasificación.

3. Algoritmos de Clasificación

Para esta parte del proyecto, se han implementado varios modelos de clasificación y se ha hecho un estudio de su rendimiento.

Todos los algoritmos han sido implementados mediante sklearn. Los algoritmos implementados han sido los siguientes: KNeighborsClassifier, LogisticRegression, LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis, GaussianNB, DecisionTreeClassifier, BaggingClassifier, RandomForestClassifier, GradientBoostingClassifier, SVC, MLPClassifier, XGBClassifier, LGBMClassifier y StackingClassifier.

Para todos los que tienen hiper parámetros se ha hecho validación cruzada.

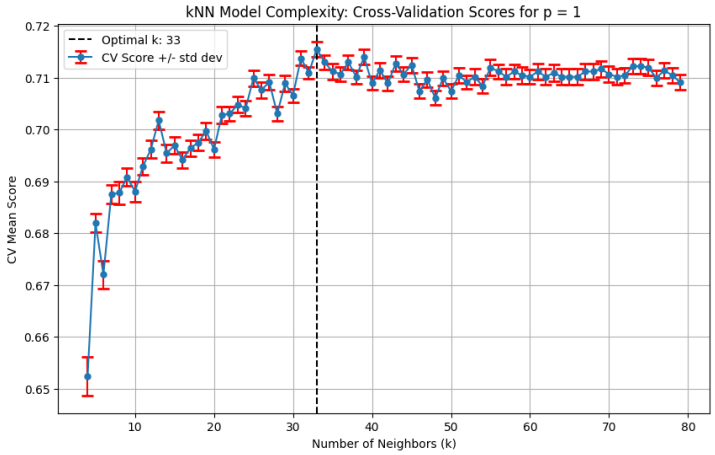

Fig. 5 Validación cruzada KNN

Los resultados obtenidos en términos de acuraccy se muestran en la figura 6.

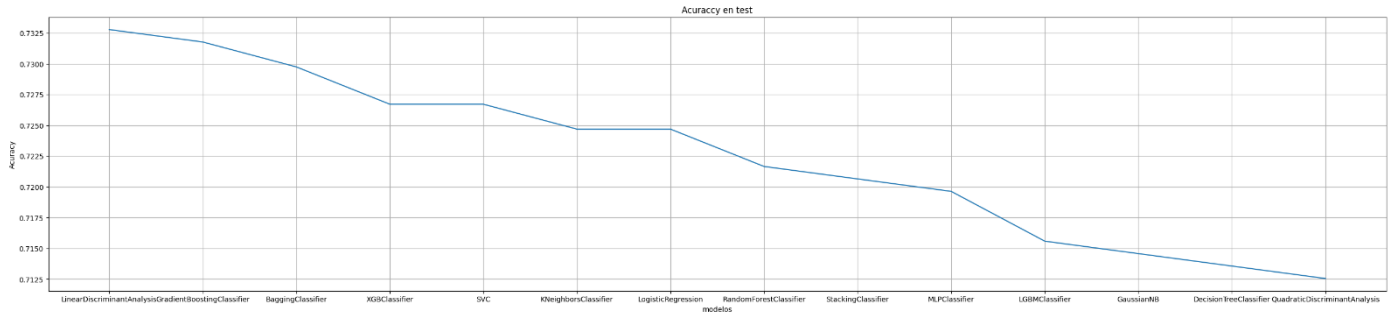

Fig. 6 Acuraccy en test para todos los modelos

También se han utilizado otras técnicas para evaluar los modelos de clasificación, como curvas ROC, matrices de confusión, curvas precisión-recall e histogramas de probabilidades predichas para los modelos.

4. Aprendizaje no supervisado

Se han aplicado 2 técnicas de aprendizaje no supervisado: PCA y Clustering.

Para PCA, se han sacado desde 1 hasta 10 componentes principales. Se ha medido el PVE y se ha entrenado un modelo de KNN. Se observa que mientras que el PVE aumenta mucho con las dimensiones, el acuraccy de KNN también pero menos. Se observa un cierto punto en el que baja, esto puede deberse a la maldición de la dimensionalidad. En este caso, convendría usar PCA para KNN, no solo por que se bajan las dimensiones, si no por que funciona ligeramente mejor.

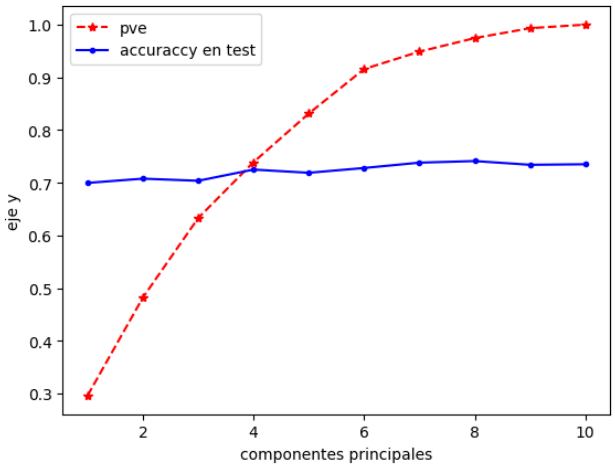

Fig. 7 PVE y acuraccy de KNN en función de las dimensiones

En cuanto al clustering, se han implementado varias técnicas: K-Means, Agnes, Birch y MiniBatchKMeans. El número de clusters óptimo se ha elegido mediante el método del codo y/o el método de la silueta.

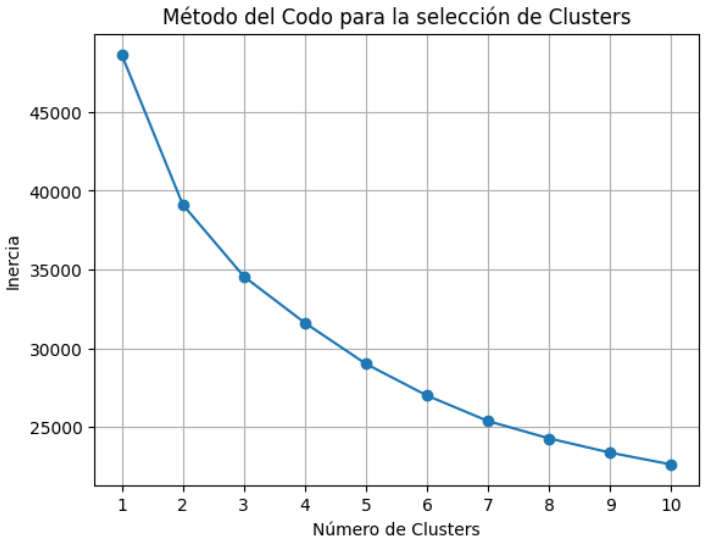

Fig. 7 Método del codo para K-means

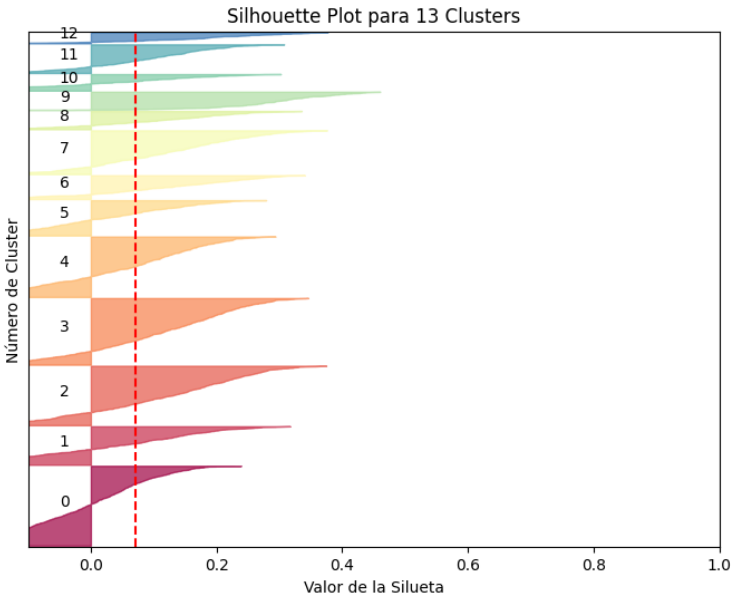

Fig 9 Método de la silueta Agnes 13 clusters (no son los óptimos)

En todos los métodos de clustering implementados se ha llegado a que el numero óptimo de clusters es 2. Que es igual al numero de clases de la variable objetivo.

5. Preguntas interesantes e hipótesis

➔ ¿Se confirman las hipótesis sobre la importancia de variables de la Parte 1 en la Parte 2?

La respuesta es sí. Las variables más importantes, vistas con el pps score, también son las más importantes al medir la importancia de las variables con Bagging y Random Forest.

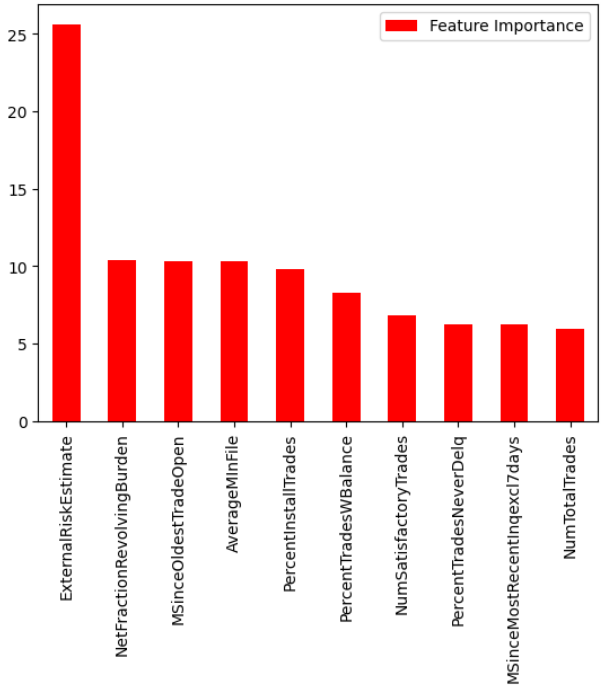

Fig. 10 Importancia de las variables Bagging

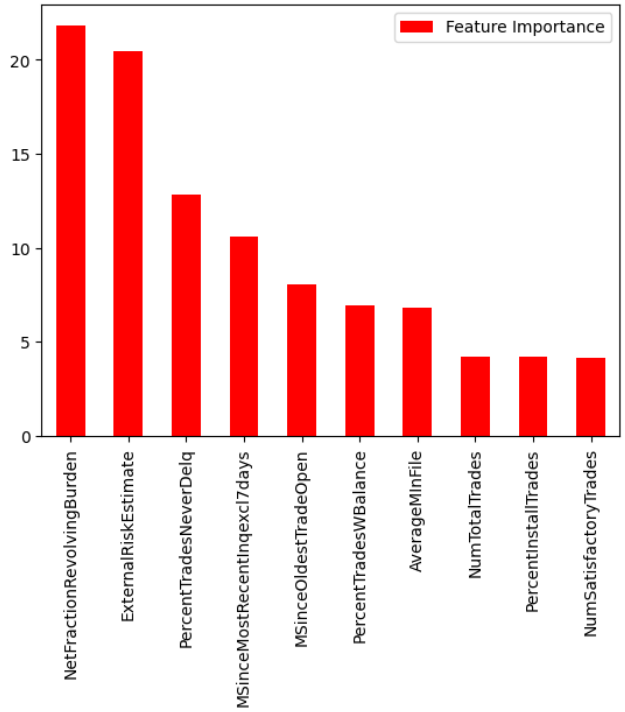

Fig. 11 Importancia de las variables Random Forest.

➔ ¿Cuál es el mejor algoritmo de clasificación? ¿Por qué?

LDA funciona muy bien. Aunque antes se ha visto en los contrastes no paramétricos que la normalidad era nula, LDA considera que los datos pertenecientes a cada clase se distribuyen como una normal. Sin embargo con un test de Anderson-Darling se llega a la conclusión de que esas distribuciones en función de la clase no son normales, lo que sorprende dada su buena performance. Además el dataset proporcionado no es muy grande y muchas de las variables tienen muy poca importancia en el resultado final (pps) (podrían considerarse como ruido). Este, muy probablemente, sea el motivo de su buen rendimiento.

En la gráfica de los acuraccys (Fig. 6) le siguen muy de cerca Boosting, Bagging y XGBoost. Estos son métodos de ensamble que capturan muy bien relaciones complejas. Tampoco sería mala idea implementarlos para este caso.

Cabe destacar que los resultados de los acuraccys dependen del split de datos que se haga, en otros splits probados, ha salido Random Forest como ganador, que es un modelo de ensamble. En general, los que funciona bien son los mencionados.

El modelo de SVC con kernel rbf también ha salido vencedor en algunos casos, esto puede deberse a la alta dimensionalidad de los datos.
En definitiva, LDA los ensambles mencionados y SVC funcionan bastante bien para este problema.
Por último, el árbol de clasificación es el que menor acuraccy en test devuelve, sin embargo tampoco funciona mal y ofrece una ventaja significativa frente al resto de modelos: la explicabilidad. Dada la naturaleza de este problema, esto puede ser muy interesante, ya que se podría necesitar saber por qué se decide lo que se decide.

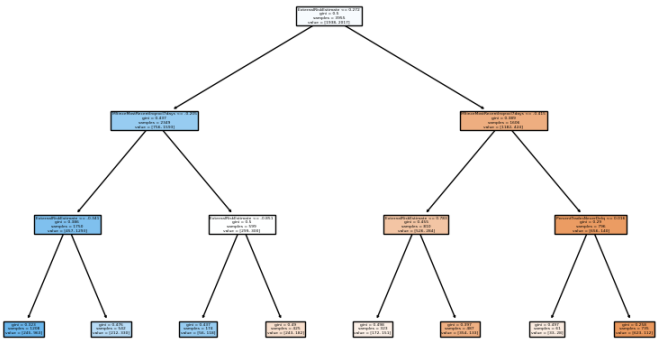

Fig. 12 Árbol de clasificación

Cabe destacar que en caso de necesitar un nivel alto de explicabilidad, aplicar PCA para reducir la dimensionalidad no sería muy buena idea debido a los cambios de base. Las dimensiones que surgen son combinaciones lineales de las dimensiones originales, y esto podría ser complicado de interpretar.

En resumen, LDA, SVC y los ensambles funcionan bien para este caso. El árbol no sería mala idea dada su explicabilidad. El mejor modelo dependería de las necesidades del cliente o de los objetivos que se quieran conseguir. Si se busca simplicidad y buen performance LDA, ya que es no paramétrico y es un modelo muy simple. Si la simplicidad no es algo que se busque, SVC con kernel rbf o los ensambles pueden ser muy buena opción. Si se requiere un alto nivel de explicabilidad (que es bastante probable para este caso) lo mejor sería un árbol.

Si se utilizan métodos de agrupamiento para la tarea de clasificación, ¿sería el rendimiento similar al de los algoritmos en la Parte 2?

Si, se obtiene un acuraccy en test en torno a 0.7, parecido a los obtenidos con el resto de modelos. No sería descabellado usarlo para clasificar.

➔ ¿Si se quitan variables poco significativas mejora la performance?

No, se ha intentado hacer eso y no ha mejorado la performance. Parece ser que aunque sean poco significativas, algo influyen en el resultado. Lo mejor que se podría hacer en este problema para reducir la dimensionalidad es PCA, que como ya se ha visto antes funciona bastante bien.

6. Bibliografía

- User guide: contents. (s. f.). [Scikit-learn](https://scikit-learn.org/stable/user_guide.html)

- Mondal, A. (2023, 17 de agosto). [Complete guide on how to Use LightGBM in Python](https://www.analyticsvidhya.com/blog/2021/08/complete-guide-on-how-to-use-lightgbm-in-python/). [Analytics Vidhya](https://www.analyticsvidhya.com/).

- What is XGBoost? (s. f.). [NVIDIA Data Science Glossary](https://www.nvidia.com/en-us/glossary/xgboost/).

- Mitchell, R. (2022, 10 de octubre). [Gradient Boosting, Decision Trees and XGBoost with CUDA | NVIDIA Technical Blog](https://developer.nvidia.com/blog/gradient-boosting-decision-trees-xgboost-cuda/). [NVIDIA Technical Blog](https://developer.nvidia.com/blog/).

- GeeksforGeeks. (2022, 20 de junio). [ML BIRCH Clustering](https://www.geeksforgeeks.org/ml-birch-clustering/).

- GeeksforGeeks. (2023, 23 de enero). [ML Mini Batch K-means clustering algorithm](https://www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/).