

# Proyecto Final: Machine Learning

Nombre estudiante: Miguel Ángel Huamani Salinas

Código estudiante: 202204727

# Índice

<b>1. Exploratory Data Analysis (EDA) . . . . .</b>	<b>3</b>
1. Tratamiento de valores nulos y especiales . . . . .	3
2. Tratamiento de valores atípicos . . . . .	4 - 5
<b>2. Classification . . . . .</b>	<b>5</b>
1. Entrenamiento y optimización de modelos . . . . .	5 - 6
2. Análisis comparativo y visual . . . . .	7
<b>3. Unsupervised Learning . . . . .</b>	<b>8</b>
1. Principal Components Analysis . . . . .	8
2. Algoritmos de Clustering . . . . .	9
<b>4. Conclusiones . . . . .</b>	<b>10</b>

# Part 1: Exploratory Data Analysis (EDA)

El primer paso es analizar los datos para obtener información valiosa que mejore el entrenamiento del modelo. Al examinar el conjunto de datos, se identifican valores nulos y especiales (-7, -8 y -9), que representan registros faltantes, operaciones no válidas o condiciones incumplidas. Para mejorar la integridad de los datos y la interpretación de los resultados, he decidido considerar estos valores especiales como faltantes. Aunque eliminar estos valores con dropna podría reducir información importante para la predicción, considero que es lo mejor para poder trabajar con datos completos (información 100% disponible). Para los valores faltantes, también se podría haber optado por un método de imputación como por media o mediana. A pesar de esto, y debido a que, eliminando valores nulos, el tamaño del dataset se reduce de 7442 a 5245 eliminando las variables de salida nulas, y luego se reduce a 4972 eliminando valores nulos, una pérdida de 2470 datos iniciales (33% aprox), puede parecer alarmante, pero debemos tener en cuenta que en realidad partimos de 5242 datos útiles, pues el resto no nos dan nada de información por carecer de un output. Por lo que perdemos unos 270 datos “útiles”, que son relativamente pocos. Además, lo importante es quedarnos con un dataset final que tenga las clases de salida balanceadas, y este el caso (2538 datos para RiskPerformance=1, y 2434 para para RiskPerformance=0).

Ya podría empezar a diseñar el modelo de predicción, pero creo que sería útil seguir analizando el dataset para tratar de eliminar el ruido presente. La reducción de ruido y atípicos puede ser clave para obtener un modelo más preciso y de mayor generalización. He usado **sns.pairplot**, para visualizar la relación entre las características respecto al output, (puntos azules clase 0, y naranjas clase 1) y **sns.heatmap**, para obtener la matriz de correlación de las variables. La gráfica del pairplot se muestra en el *Apéndice A* y la gráfica de la matriz de correlación es la siguiente:

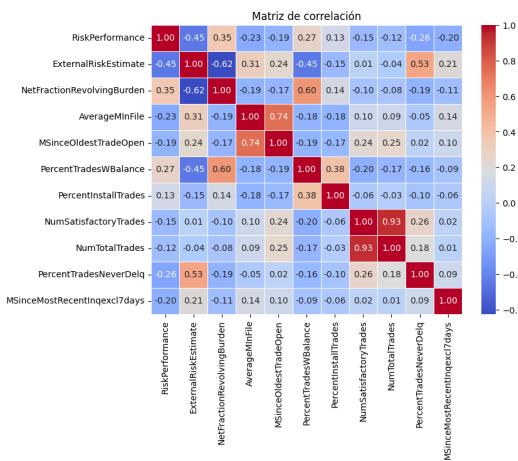


Figure 1. Matriz de correlación del dataset

El pairplot revela la relación entre el output y los predictores individuales. Aunque las distribuciones de las clases están superpuestas en la diagonal principal, la matriz de correlación puede revelar interacciones entre variables. ExternalRiskEstimate muestra una distribución de clases más separada y correlacionada con la salida. Sus nubes de puntos tienden a ser más linealmente separables, lo que me lleva a concluir que esta variable es importante a la hora de predecir, y tendré esto en cuenta para el análisis posterior.

Lo último a destacar de la gráfica es la presencia de puntos alejados que a primeras parecen valores atípicos, y que pueden aportar ruido. A modo de reducir esto, he decidido estudiar la distribución individual de cada predictor. Las distribuciones varían como se observa a continuación, por lo que dependiendo del contexto un valor será considerado atípico o no.

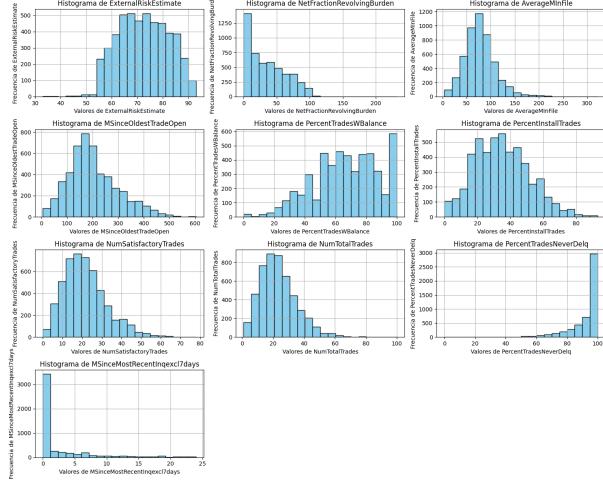


Figure 2. Histogramas de las variables de entrada.

**ExternalRiskEstimate:** Medida de riesgo externa (Estimación de que tan probable es que un cliente cumpla con el pago), y los atípicos supondrían una estimación extrema por parte de estas fuentes. La siguiente gráfica muestra la distribución de clases:

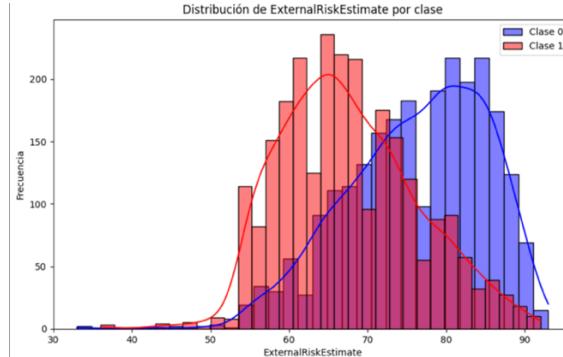


Figure 3. Distribución de clases (ExternalRiskEstimate)

Las estimaciones rondan el rango (50-90), pero cabe destacar la equivocación de una estimación baja que resultó ser de la clase 0 (el usuario cumplió con el pago). Es por esto por lo que en esta variable he preferido eliminar dicho dato que se trata de una “muy mala predicción” (la estimación a eliminar está por debajo de 35).

**NetFractionRevolvingBurden:** Proporción del saldo pendiente en las tarjetas de crédito con respecto al límite de crédito total disponible, que podría indicar la capacidad de manejo de deuda. No conozco la escala exacta, cualquier valor por encima del 100% no tendría sentido (a no ser que se permita usar más dinero del permitido). Como en la nube de puntos, que enfrenta esta variable al resto, se observa este punto muy alejado del resto consideraré un atípico cualquier valor por encima de 200.

**AverageMInFile:** Duración promedio de las operaciones en el historial crediticio indica la estabilidad y longitud de este. Valores más altos sugieren mayor estabilidad y menor riesgo crediticio. Conservar valores altos de esta variable puede ayudar a distinguir a los prestatarios más confiables.

**MSinceOldestTradeOpen:** Antigüedad del historial crediticio de los prestatarios, reflejando la estabilidad financiera y responsabilidad crediticia. Valores más altos sugieren una mayor estabilidad. Sin embargo, valores extremadamente altos, especialmente por encima de 500 meses, se consideran atípicos y se eliminan del análisis, ya que pueden indicar casos poco comunes de incumplimiento.

**PercentTradesWBalance:** Porcentaje de cuentas de crédito activas con saldo pendiente, en comparación con el total de cuentas. No considero que hayan valores atípicos, ya que está expresado como un porcentaje entre 0 y 100.

**PercentInstallTrades:** Porcentaje de cuentas con pagos fijos periódicos, como préstamos de coches, respecto al total de cuentas. Mismo razonamiento que la variable anterior, no considero que haya atípicos, no se puede simplificar más esta variable.

**NumSatisfactoryTrades** indica el número de transacciones crediticias en las que el prestatario ha cumplido satisfactoriamente con sus obligaciones. Mientras que **NumTotalTrades** simplemente cuenta el número total de transacciones. Ambas están altamente correlacionadas (0.93), y eliminar los valores atípicos de la primera (por encima de 80 en la nube de puntos) equivale a hacerlo también en la segunda (por encima de 100). Considero que es una buena opción eliminar estos valores atípicos.

**PercentTradesNeverDelq:** indica el porcentaje de transacciones crediticias de un individuo que nunca han experimentado morosidad. Aunque todos los datos son relevantes, esta variable no es tan crucial como otras en el contexto del proyecto.

**MSinceMostRecentInqexcl7days:** representa el tiempo en meses desde la última consulta de crédito, excluyendo los últimos 7 días. No hay valores atípicos significativos, pero su contribución al modelo es limitada, como se puede inferir de la distribución de los datos en el pairplot.

Tras limpiar el dataset y eliminar el ruido, quedan 4944 datos, con 2527 en la clase 1 y 2417 en la clase 0, creando un conjunto de datos equilibrado y apto para entrenar el modelo. Ahora, la elección del modelo adecuado será crucial para obtener los mejores resultados dadas las características del proyecto.

## Part 2: Classification

He optado por utilizar modelos de clasificación que funcionan bien con clases linealmente separables, como Regresión Logística y SVM con kernel lineal, así como modelos que no, como Bagging, Random Forest y SVM con kernel no lineal. Incluiré Naive Bayes para explorar la suposición de independencia condicional entre las variables. Para simplificar el proceso, utilizaré la biblioteca scikit-learn en lugar de programar los modelos desde cero. Sin embargo, para el análisis comparativo, me

parece correcto el uso de código empleado en prácticas para visualizar el rendimiento de los modelos y ajustar los hiperparámetros, como el uso de `utils.py` para Regresión Logística. A excepción de los métodos de ensamble, el resto de los modelos son sensibles a la escala de las características por lo que he normalizado los datos con el uso del **StandardScaler**, después de haber hecho la división train/test. Todos los modelos serán almacenados en un diccionario, para obtener un dataframe con las métricas de estos, y una visualización del performance de cada uno. Para logistic regression, la mejor precisión se obtiene con la regularización Elastic Net (y un L1\_ratio de 0.1). La obtención del C óptimo (1) se ha obtenido como en la práctica:

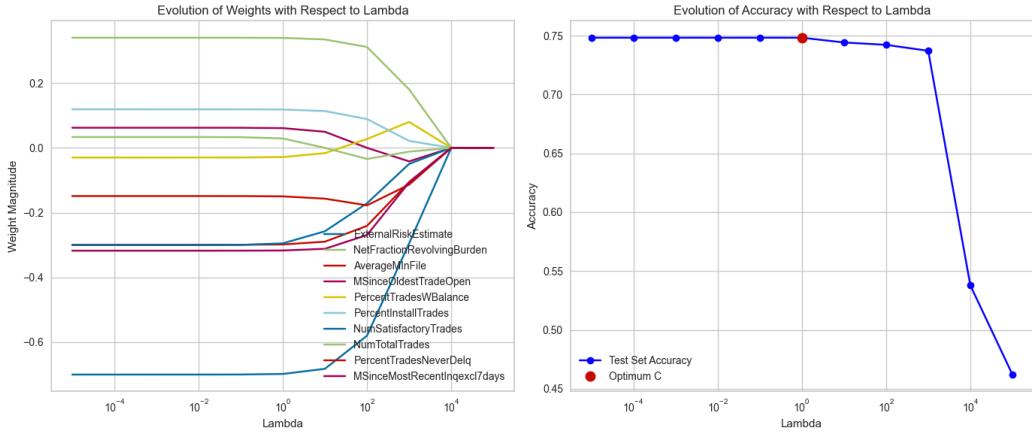


Figure 4. Evolución de pesos y accuracy para valores de C.

De la gráfica de evolución no se puede destacar ningún peso que vaya a 0 antes, si no que todos parecen tender a 0 cuando C vale alrededor de  $10^4$ . Para Naive Bayes no he utilizado ningún método de generalización para los datos, y para el resto de los modelos he utilizado GridSearchCV para hacer validación cruzada de la combinación de distintos hiperparámetros. Para Naive Bayes he decidido quedarme únicamente con las métricas, y no dibujar algún tipo de gráfica, mientras que para el método de ensamble Bagging, si he considerado que lo mejor era mostrar el histograma de la importancia de variables (veces que ha sido usada para un corte). El uso de este modelo ha sido principalmente para ver esta importancia de variables más que para entrenarlo, ya que un problema con la librería me impide almacenar el modelo de Bagging óptimo en mi diccionario de posibles modelos.

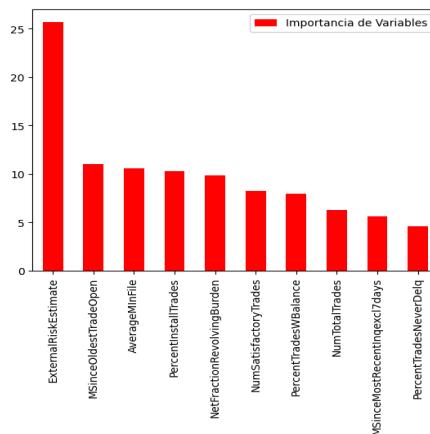


Figure 5. Importancia de variables (Bagging)

Del histograma se puede observar como las variables de menor importancia son aquellas que contextualmente no parecían influenciar mucho. Para SVM, si he decidido explorar de forma visual la combinación de hiperparámetros con validación cruzada. Decidí crear dos filas de gráficos, donde cada fila corresponde a un kernel y cada columna corresponde a un valor de C (gráfica en el código y en el *Apéndice B*), obteniendo finalmente un modelo óptimo para cada kernel. Finalmente he decidido visualizar el performance de cada modelo en mí conjunto de train con dos gráficas principales: la ROC Curve y la Calibration Curve:

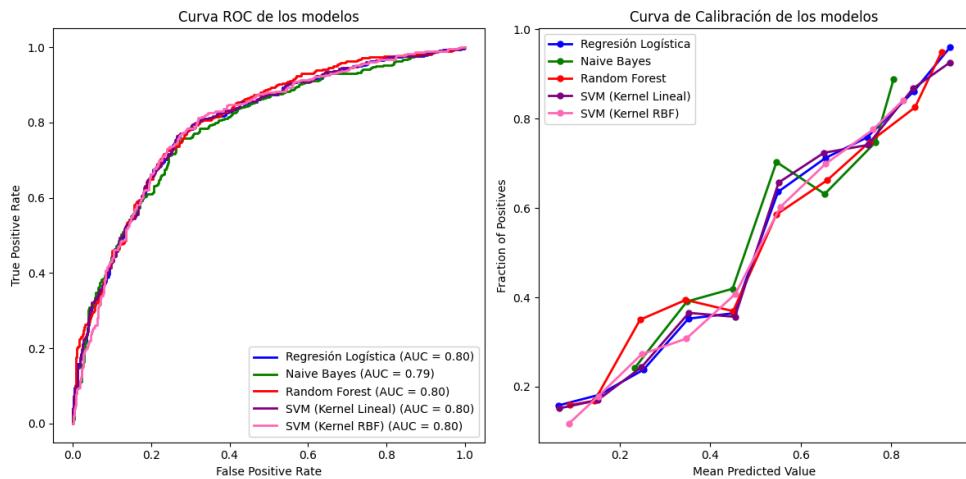


Figure 6. Curvas ROC y Calibration Curves de mis modelos

Se puede observar como las 5 curvas ROC, tienen una AUC idéntico si no muy similar de 0.8, llevándome a pensar que todos los modelos están funcionando relativamente bien en términos de discriminación. En cuanto a las curvas de calibración, Logistic Regression y SVM, son los modelos que más se asemejan a la recta principal por lo que las probabilidades predichas por estos dos modelos están un poco mejor calibradas y son más consistentes con la frecuencia real de ocurrencia de las clases, que los otros modelos. Para poder analizar mejor el performance de cada modelo, he decidido mostrar el siguiente datafame con las métricas de todos los modelos ya optimizados.

	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.748231	0.764486	0.768797	0.766635
Naive Bayes	0.729019	0.761905	0.721805	0.741313
Random Forest	0.747219	0.756364	0.781955	0.768946
SVM (Linear Kernel)	0.749242	0.765918	0.768797	0.767355
SVM (Radial Kernel)	0.746208	0.758748	0.774436	0.766512

Figure 7. Dataframe de las métricas de mis modelos

Del datafame se ve que todos los modelos tienen una precisión (accuracy) similar, alrededor del 74-75% (a excepción de Naive Bayes), sugiriendo que todos los modelos tienen un rendimiento similar en la clasificación. Los modelos que destacan son SVM lineal y Logistic Regression, llevándome a pensar que las clases podrían ser linealmente separables en el espacio de características, dentro de un margen aceptable de error. A pesar de esto, y debido a que modelos no lineales como Random Forest y SVM radial

también muestran tener buenas métricas, no creo poder deducir con exactitud la razón del modelo óptimo, y creo que todavía existe un margen para poder mejorar el rendimiento de los modelos.

## Part 3: Unsupervised Learning

El propósito principal del análisis de PCA es la reducción de la dimensionalidad de los datos conservando a la vez la mayor cantidad posible de información. Con los datos ya estandarizados, lo primero que he hecho ha sido graficar la acumulación de varianza con el fin de poder observar que número de componentes principales me capturará la varianza suficiente y así deshacerme del resto.

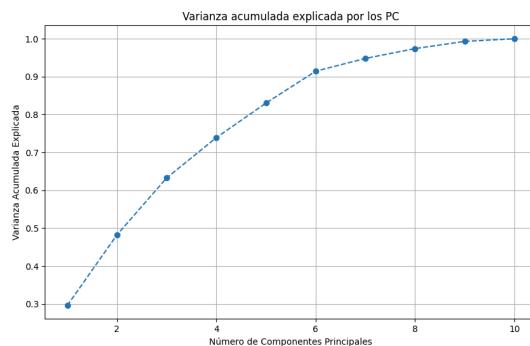


Figure 8. Varianza acumulada de los componentes principales

La gráfica sugiere que las primeras 5 componentes principales son suficientes para capturar la varianza del dataset. Los loadings en el *Apéndice C* muestran la contribución de cada característica original a los primeros dos PC's. Por ejemplo, PC1 parece relacionarse con el manejo de la deuda pendiente y la estabilidad financiera, mientras que PC2 refleja la variedad de cuentas crediticias y la diversificación del riesgo. La siguiente tabla muestra el performance de mis modelos de clasificación en este nuevo dataset de dimensión reducida a 5 componentes principales.

Modelos	Accuracy
Logistic Regression	0.7426
Naive Bayes	0.7350
Random Forest	0.7290
SVM (lineal)	0.7431
SVM (RBF)	0.7462

Figure 9. Accuracy de mis modelos en el espacio de PC1 y PC2

Como se puede observar en la tabla la reducción de dimensiones no ha hecho que se pierda tanta precisión como podría esperarse, todo los accuracies rondan el 72-75%, por lo que considero que la reducción de dimensiones es una buena práctica si queremos simplificar los datos. A modo de visualizar lo obtenido, aquí se muestran ambas clases en este espacio de dimensiones reducido para dos componentes principales (ya que no puedo visualizarlo para 5, que es lo que más sentido tendría).

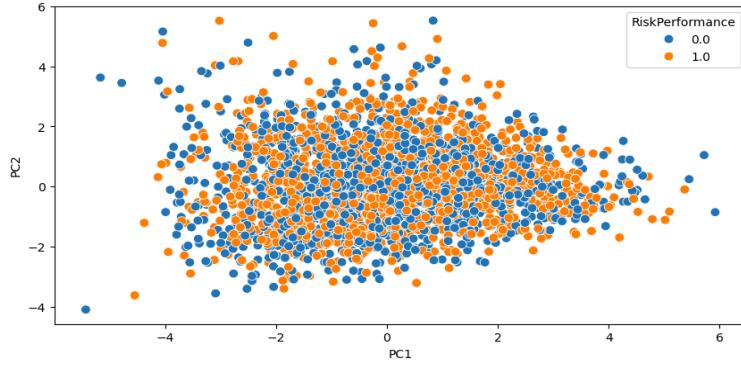


Figure 10. Clases en la dimensión reducida (PC1, PC2)

He decidido que para visualizar el agrupamiento de los clusteres usaré este nuevo dataset de las dos primeras componentes principales. Para este apartado he decidido hacer uso únicamente de dos algoritmos de clustering: KMeans y Agglomerative clustering (un tipo de Hierarchical clustering). A modo de identificar el numero óptimo de clústers, he utilizado tanto el método del codo como el de la silueta, como se muestra a continuación para el algoritmo KMeans.

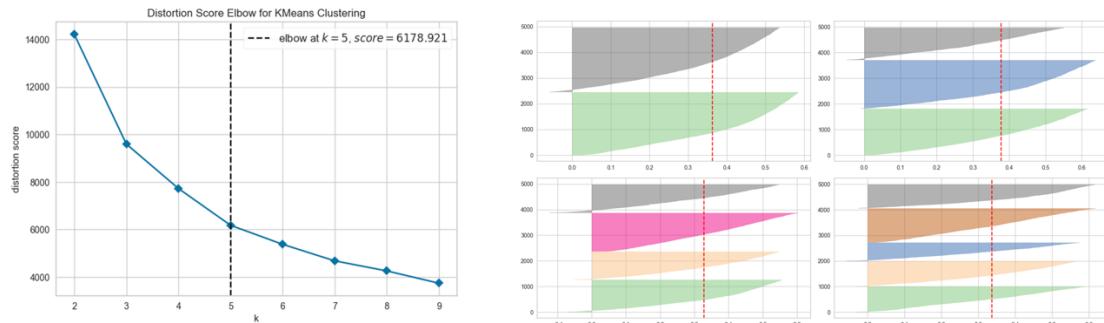


Figure 11. Método del codo y de la silueta.

De las gráficas se puede observar que la silueta media máxima se obtiene con 3 o 2 clústeres, mientras que con el método del codo es con 5. Lo mismo ocurre con Agglomerative, ya que se obtienen 3 y 2 con el método de la silueta, y 4 con el del codo como se puede observar en el Apéndice C. He decidido quedarme con el mayor número de clusters (5 y 4) en ambos casos obteniendo el siguiente resultado en mí nuevo espacio de dimensiones reducido.

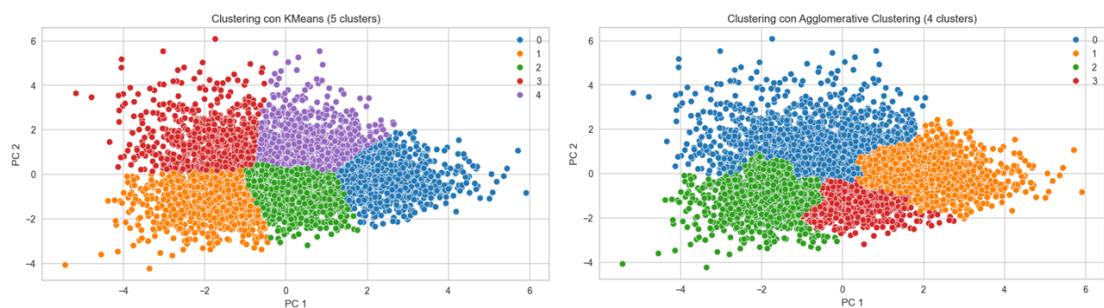


Figure 12. Clustering con KMeans (5 clusteres) y con Agglomerative (4 clusteres)

## Part 4: Conclusions

Analizando lo obtenido, destaco la importancia de ciertas variables de entrada. La exploración contextual realizada en la Parte 1 proporcionó una idea inicial sobre qué variables podrían ser más relevantes en el proyecto, y esto se confirmó en la Parte 2. Por ejemplo, se observó que la variable ExternalRiskEstimate es la más importante, mientras que PercentTradesNeverDelq y MSinceMostRecentInqexcl7days son las menos relevantes (estas dos variables no aportan información útil para predecir impagos). Además, NumTotalTrades también tiene poca importancia, posiblemente debido a su alta correlación con NumSatisfactoryTrades. Eliminar estas tres variables podría reducir la precisión del modelo, porque aun teniendo poca importancia, pueden ser empleadas con otras, pero generalizaría mejor el modelo.

En cuanto al mejor algoritmo de clasificación, no existe una respuesta certera ya que como se pudo observar en la Parte 2, varios modelos muestran tener una performance medianamente buena en el conjunto de prueba. SVM con kernel lineal y Regresión Logística parecen ser mejores, aunque SVM con kernel radial y Random Forest también ofrecen resultados similares para la separación de clases no lineales. Naive Bayes funciona bastante bien, por lo que no es un mal enfoque considerar la independencia condicional de características. Dada esta variedad y considerando la importancia de la varianza, se realizará una validación cruzada en todo el conjunto de datos para evaluar las precisiones y varianzas de cada modelo. El objetivo es encontrar un equilibrio entre baja varianza y alta precisión para seleccionar el modelo más adecuado.

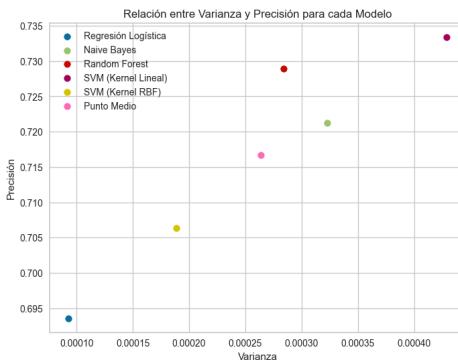
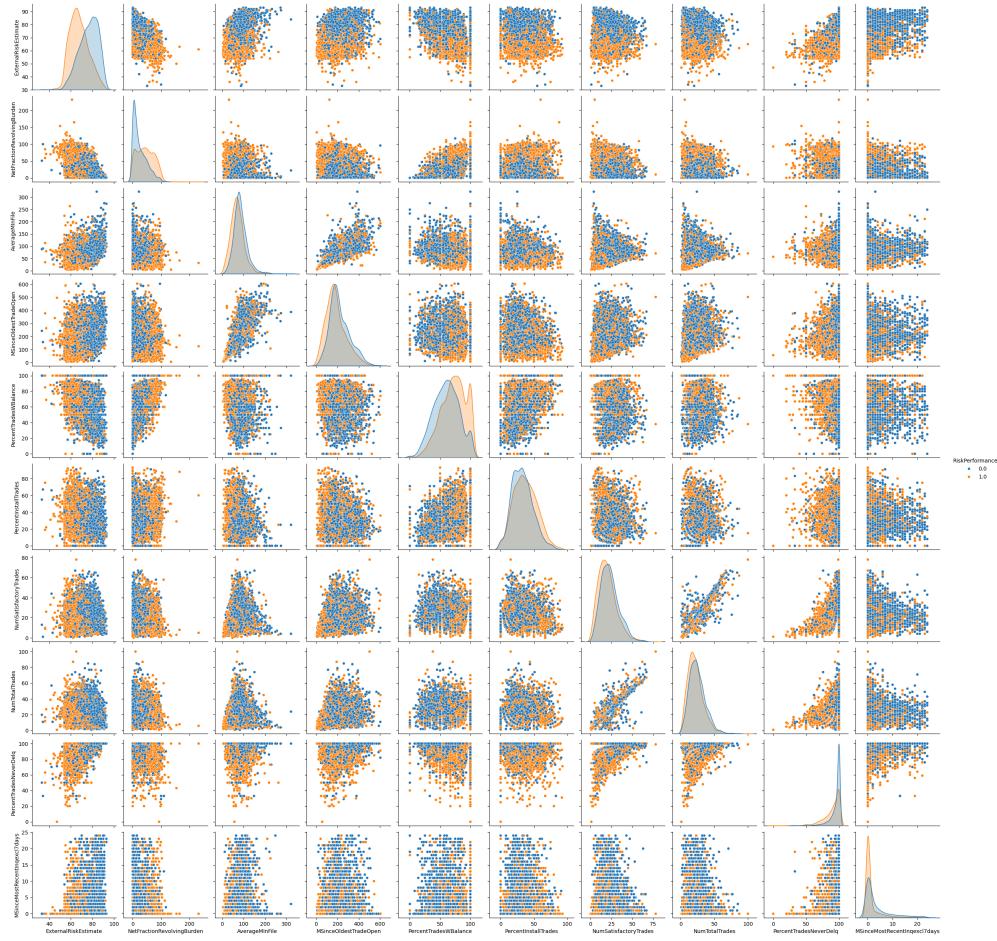


Figure 13. Accuracy y varianza de los modelos con validación cruzada del dataset.

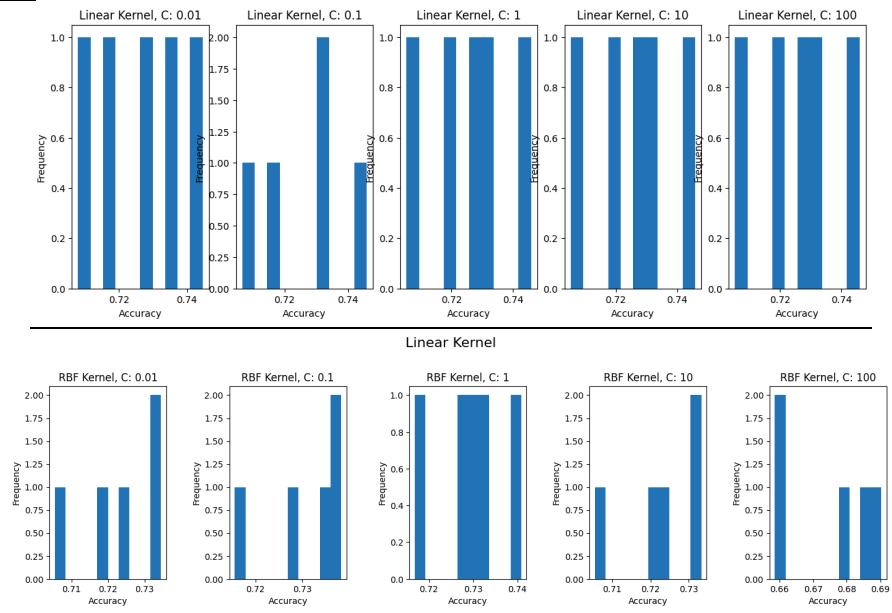
La gráfica sugiere que el modelo más estable sería Logistic Regression, ya que muestra la menor variabilidad, aunque menor precisión. SVM lineal presenta mayor variabilidad, pero también una precisión más alta. Sin embargo, la diferencia en la variabilidad entre los modelos es mínima, lo que sugiere que la diferencia práctica en la estabilidad es insignificante. En cuanto a las métricas, un modelo ideal debería tener alto recall para capturar la mayoría de los préstamos de riesgo, manteniendo una alta precisión para evitar clasificar incorrectamente. Dado que Random Forest se encuentra cerca del punto medio y tiene una alta precisión, es mi elección como modelo final. Finalmente, los algoritmos de clustering utilizados en la Parte 3 buscan patrones en los datos mediante la agrupación basada en similitudes, sin tener en cuenta las etiquetas de clase. Pueden identificar estructuras en los datos que no necesariamente coinciden con las reales. Al comparar las clases en un espacio de dimensiones reducido con los clusters determinados por los algoritmos, se observa que las clases están muy mezcladas en la nube de puntos, lo que resultaría en una mala clasificación.

# Apéndice

## Apéndice A



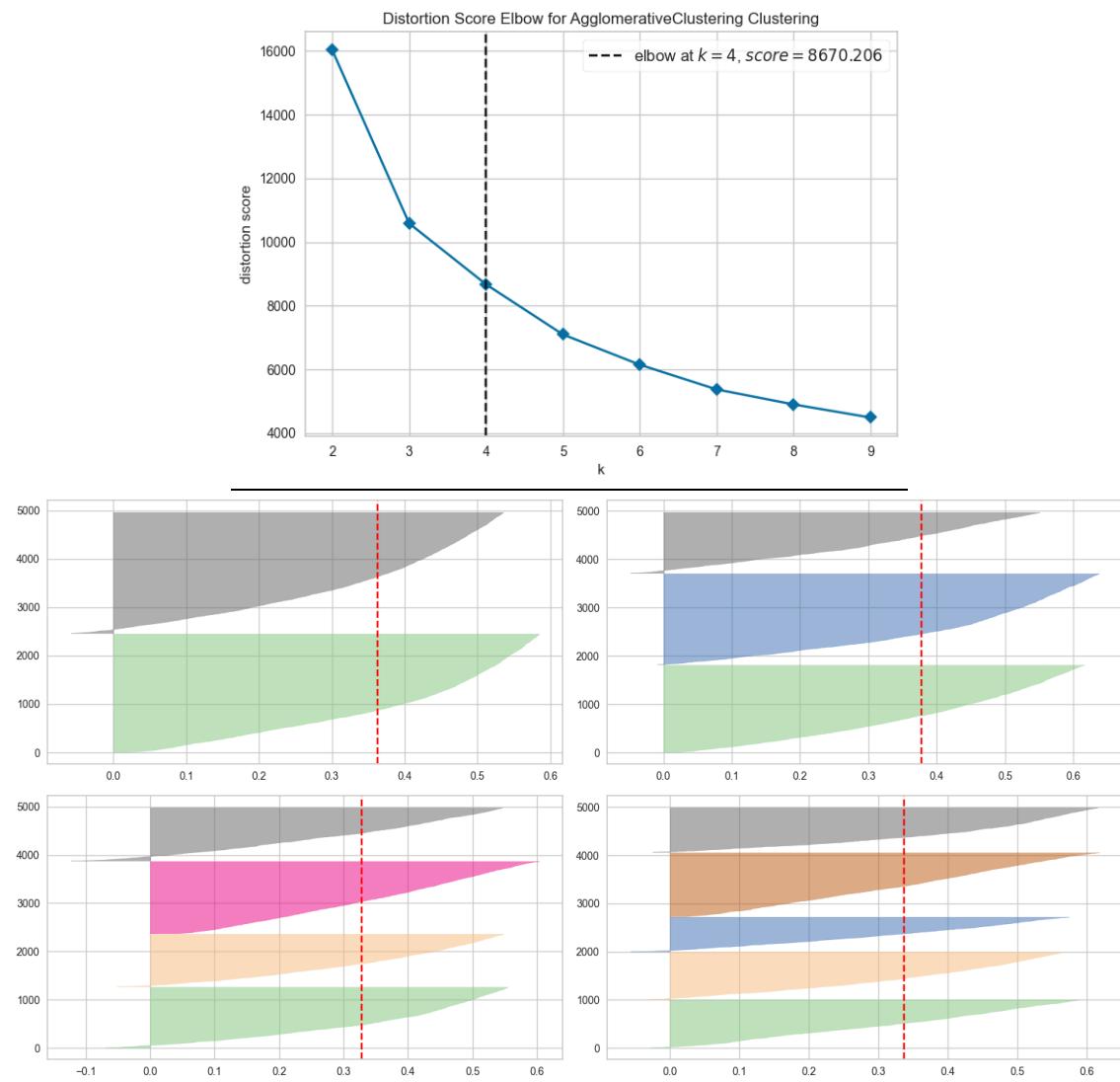
## Apéndice B



## Apéndice C

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
P C 1	-0.409	0.38 8	-0.322	-0.345	0.40 2	0.22 4	-0.289	-0.269	-0.257	-0.149
P C 2	-0.344	0.26 9	-0.046	0.088	0.16 0	0.11 5	0.601	0.618	-0.006	-0.124

## Apéndice D



## Bibliografía

- *Selecting the number of clusters with silhouette analysis on KMeans clustering.* (s. f.). Scikit-learn. [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html#sp\\_hx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html#sp_hx-glr-auto-examples-cluster-plot-kmeans-silhouette-analysis-py)
- *Sklearn.decomposition.PCA.* (s. f.). Scikit-learn. <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
- *Sklearn.svm.SVC.* (s. f.). Scikit-learn. <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>