UNIVERSIDAD DE ANTIOQUIA

MODELO PREDICTIVO PARA EL APOYO A LA GESTIÓN DE LA CARTERA DE EMPRESA ADMINISTRADORA DE RIESGOS LABORALES

Autor(es)

Ivonne Ortega Echeverry

Diana Milena Toro Arrieta

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas

Medellín, Colombia

2021

Modelo predictivo para el apoyo a la gestión de la cartera de empresa administradora de riesgos laborales

Ivonne Ortega Echeverry
Diana Milena Toro Arrieta

Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de:
Especialista en Analítica y Ciencia de Datos

Asesores (a):
Daniela Serna Buitrago, Especialista en Analítica

Universidad de Antioquia
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas
Medellín, Colombia
2021

TABLA DE CONTENIDOS

1. RESUMEN EJECUTIVO

Las Administradoras de Riesgos Laborales o ARL, son entidades aseguradoras de vida y cubren riesgos de tipo laboral, tanto accidentes como enfermedades laborales. De carácter obligatorio para empresas y trabajadores autónomos y es regulada por el estado.

El negocio de la ARL depende en gran parte del recaudo de la cotizaciones que los trabajadores independientes, prestadores de servicio y/o las empresas realizan por cada uno de sus empleados, para la inversión en programas de prevención en el trabajo y pagos por accidentes, enfermedades, incapacidades pensiones de invalidez o cualquier otra condición derivada de la actividad laboral. Por lo que una adecuada gestión anticipada de la cartera se hace evidente dentro del proceso de recaudos de la empresa. Y es así que, como apoyo a la necesidad evidenciada en conversaciones con el negocio nace este proyecto, el cual busca predecir qué empresas afiliadas a la ARL tendrán mayor probabilidad de quedar en estado de mora el mes inmediatamente posterior a la consulta.

Para lograr este objetivo se pidió permiso a la empresa para el uso de los datos con fines académicos, estos datos previa extracción y por su sensibilidad, debieron ser anonimizados y solo se trabaja con valores estadísticos y labels numéricos que no permitan el reconocimiento de las empresas dentro del dataset, se buscó además, que se incluyera datos de pre-pandemia y pandemia, dado que por efectos del pandemia generada por el COVID-19, las empresas tienen comportamientos muy atípicos a los esperados en años anteriores.

En cuanto al modelado en la fase de limpieza, se encuentra con un dataset bastante completo, por lo que solo se requirió hacer limpieza de unos pocos datos nulos y cambios datos tipo fecha por tipo numérico, resultado de las diferencias de estos campos tipo date, que se consideran formatos más sencillos de operar y brindan más información.

Ya en la fase de implementación y ejecución de modelos, la estrategia a seguir es primero generar una línea base o *Baseline*, con una primera corrida de varios modelos sin modificación de hiperparametros o adición de alguna técnica de ingeniería de características, que permita tener una base contra la cual comparar el resultado de modelos más robustos. Y posteriormente realizar corridas de varios algoritmos buscando de los mejores hiperparametros, junto con técnicas para el tratamiento de datos desbalanceados que permitan mejorar el *score* del mejor modelo obtenido en el *Baseline.*

Se toma el resultado del algoritmo *Easy Ensemble* como *baseline, dado* que su *score* de precisión, aunque fue el más bajo en esa etapa, su matriz de confusión muestra que es el que menos se equivoca en la clasificación, posteriormente en una búsqueda de hiperparametros, se encontrará que el algoritmo *RandomForestClasificator* con un *score* de precisión de 70% logra clasificar mejor las muestras y adicional que todos los intentos de generación de muestras sintéticas y agregación de datos de fuentes externas solo logran desmejorar el resultado obtenido por la mejor combinación de hiperparametros encontrados con el método *gridShearch.*

El repositorio del proyecto: https://github.com/anaidg/Seminario-Gestion_Cartera.git

2. DESCRIPCIÓN DEL PROBLEMA

Las Administradoras de Riesgos Laborales o ARL, son entidades aseguradoras de vida y cubren riesgos de tipo laboral, como accidentes y enfermedades laborales. Y cuyo producto (el seguro) es de carácter obligatorio para todas las empresas y trabajadores autónomos, por lo que están reguladas por el estado.

El negocio de la ARL depende del recaudo de la cotizaciones que los trabajadores independientes, prestadores de servicio y/o las empresas realizan por cada uno de sus empleados, para la inversión en programas de prevención en el trabajo y pagos por accidentes, enfermedades, incapacidades pensiones de invalidez o cualquier otra condición derivada de la actividad laboral. Por lo que una adecuada gestión anticipada de la cartera se hace evidente dentro del proceso de recaudos de la empresa, por lo que se quiere detectar aquellas empresas que según su comportamiento podrían llegar a caer en mora, y así crear planes que ayuden a mitigar el riesgo.

2.1 PROBLEMA DE NEGOCIO

Después de conocer el proceso de recaudos de la ARL, se detecta una oportunidad de mejora al proceso, esto porque, según lo indicado por el negocio, los pagos a la ARL se realizan mes vencido, por lo que el cierre contable del mes actual corresponde al mes inmediatamente anterior y eso hace que los análisis de la información y la toma de decisiones se hagan con datos de aproximadamente 2 meses hacia el pasado y no se tiene una gestión oportuna sobre temas de cartera, dado que la realidad de las empresas afiliadas puede llegar a variar mes a mes, según el sector económico, época del año, inflación, cambio en el valor del dólar, entre otras.

Es así que, como apoyo a la necesidad evidenciada en conversaciones con el negocio nace este proyecto, el cual busca predecir qué empresas afiliadas a la ARL tendrán mayor probabilidad de quedar en estado de mora el mes inmediatamente posterior a la consulta y que de esta manera tener lograr tener una gestión más oportuna ante el riesgo de mora que puedan presentar las empresas afiliadas.

2.2 APROXIMACIÓN DESDE LA ANALÍTICA DE DATOS

Dado que se cuenta con los datos históricos, demográficos, valores y fechas de pago, entre otros. Se pretende hacer uso del conocimiento de modelos y algoritmos analiticos predictivos, que permitan predecir la probabilidad de morosidad de las empresas afiliadas. Con un nivel de

confianza aceptable para el negocio, el cual será determinado por la empresa a medida que se vaya validando los resultados del modelo en meses posteriores a su salida a producción, dado que actualmente no se cuenta con ejercicios previos parecidos contra los cuales el negocio pueda dar una métrica de valor esperado.

2.3 ORIGEN DE LOS DATOS

EL dataset proporcionado para el proyecto académico tiene como fuente el *Data WareHouse* de la compañía, consta de 30 variables, incluida la etiqueta de morosidad. Dada la regulación interna de los datos y que esta debe poder ser compartida en un repositorio externo a la compañía se nos permite tomar solo 6 meses de información para el entrenamiento, donde el periodo de tiempo seleccionado para la extracción de los datos fue 2020-01-01 al 2020-30-06, lo que nos permite tener datos de las empresas antes y durante la pandemia, con esto se busca que los modelos entrenados respondan a los 2 escenarios, dado que el 2020 y lo que va del 2021 son considerados años atípicos y el comportamiento de las empresas se ha visto directamente afectado por el entorno.

Los datos constan de variables demográficas y de segmentación interna, así como también, valores de pago y saldos pendientes, se buscó que las mismas empresas estuvieran a lo largo de los 6 meses escogidos, para buscar posibles patrones que demuestren temporalidad en las variables.

3. DATOS

3.1 DATOS ORIGINALES

La empresa brinda un data set que consta de 30 variables, incluida el marcador de morosidad, del periodo de tiempo de 2020-01-01 al 2020-06-01 y la fuente de la información es el Data WareHouse de la compañía.

Donde se cuenta con 1057797 registros únicos, en un archivo en formato csv y con un peso 250 Mg aproximadamente.

Las variables del dataset son:

nombre del campo	**tipo**	**descripción**
Id	integer	identificador único de la empresa
Mes_id	integer	Identificador año-mes

Uen_Arp_Id	integer	identificador(codigo) UEN numerico
Regional_Arp_Id'	integer	identificador (código) de la regional numérico
Oficina_Arp_Id'	integer	identificador (código) de la oficina numérico
Tamano_Empresa_Arp_Id	integer	identificador (código) del tamaño de numérico
Estado_Empresa_Arp_Id	integer	marca- indicador de mora o en cobertura de las empresas
cant_siniestros	integer	número de siniestros asociados a un contrato
Sector_Economico_Id	integer	identificador (codigo) del sector economico numerico
clasificacion_riesgo	integer	identificador (código) para clasificar el nivel de riesgo de la actividad económica numérico
Fecha_Inicio_Cobertura	date	fecha en que la empresa indica que el empleado inicia cobertura
ultimo_dia_mes	date	fecha último día del mes, correspondiente al mes_id
Nro_afiliados	integer	cantidad de afiliados o trabajadores por empresa
Fecha_Limite_Pago	date	fecha límite de pago asignada a la empresa
fecha_real_pago	date	fecha en que se realiza el pago por parte de la empresa
avg_Valor_Salario	decimal(17,4)	media de los salarios de los trabajadores
IBC	decimal(17,4)	Ingreso Base de Cotización
Numero_Dias_Esperados	integer	días esperados para ser pagados
Numero_Dias_Reportados	integer	los días que se reportan en la planilla del pago
Numero_Total_Coberturas	integer	número de coberturas del contrato
Valor_Cotizacion_Esperada	decimal(17,4)	cotización calculada
Valor_Cotizacion_Reportada	decimal(17,4)	cotización reportada
Valor_Ingreso_Base_Liq_Esp	decimal(17,4)	totalizado Ingreso Base liquido esperado
Valor_Ingreso_Base_Liq_Rep	decimal(17,4)	totalizado Ingreso Base liquido reportado
Valor_Saldo'	decimal(17,4)	saldo pendiente en el mes
Valor_Tasa_Cotizacion_Esp	decimal(17,4)	totalizado valor tasa cotización esperado
Valor_Tasa_Cotizacion_Rep	decimal(17,4)	totalizado valor tasa cotización esperado
valor_cartera	decimal(17,4)	saldo en cartera mes actua
valor_cartera_mes_ant	decimal(17,4)	valor del saldo en cartera mes anterior

en_cartera	char(1)	indicador de la empresa se encuentra marcada en mora 0 = NO / 1=SI

Los datos quedan en el repositorio de drive asignado a cada estudiante por la universidad, para acceder a estos se debe generar una solicitud a Diana Toro, diana.toro@udea.edu.co.

3.2 DATASETS

El archivo entregado es cargado en un *DataFrame,* y a partir de ahí comenzar a conocer los datos y realizar limpieza, la cual consiste en el borrado o imputación de datos y/o registros que nos garanticen completitud, certeza e información, por lo que se comienza con la búsqueda campos nulos, registros duplicados y/o caracteres extraños

```
1 df = pd.read_csv('/content/drive/Shareddrives/Monografia/dataset_esp.txt', delimiter = "\t")
2 df.head(5)
```

	id	Mes_Id	Uen_Arp_Id	Regional_Arp_Id	Oficina_Arp_Id	Tamano_Empresa_Arp_Id	Estado_Empres
0	217684039	202002	45	20	21	5347	EN CC
1	218069202	202006	45	20	21	5347	EN CC

Imagen 1. Carga del archivo al DataFrame

```
1 #busqueda de nulos
2 df.isnull().values.any()
```

```
False
```

```
[ ] 1 #busqueda de registros duplicados
    2 dupl = df.duplicated()
    3 print(dupl.sum())
```

```
0
```

Imagen 2. Búsqueda de registros nulos y duplicados

Se observa que existe un campo con el símbolo ‘?’, en campos tipo fecha, el cual validando contra la fuente hace referencia a campos nulos, por lo que se toma la decisión de hacer el borrado del registro que contenga este símbolo dado que no se cuenta con criterio para imputarlos, en total se borran 56545 registros como se observa en la imagen 3.

```
[ ]  1 # Verificar si existe algún '?' en la columna 'Fecha_Limite_Pago'
     2
     3 ind1= df[df['Fecha_Limite_Pago'] == '?'].index
     4 ind1
```

```
Int64Index([], dtype='int64')
```

```
[ ]  1 # Verificar si existe algún '?' en la columna 'fecha_real_pago'
     2
     3 ind2 = df[df['fecha_real_pago'] == '?'].index #indica la posición del registro con '?'
     4 print('cantidad de registros con valor ? ='+ str(ind2.value_counts().count()))
     5 ind2
```

```
cantidad de registros con valor ? =56545
Int64Index([      9,      11,      14,      42,      56,      83,      92,
                 97,     105,     108,
            ...
            1057634, 1057647, 1057659, 1057662, 1057674, 1057681, 1057691,
            1057734, 1057745, 1057746],
           dtype='int64', length=56545)
```

```
[ ]  1 # Eliminar valores "faltantes" en la columna 'fecha_real_pago'(para la línea base).
     2
     3 df.drop(ind2, inplace=True)
```

Imagen 3. Busqueda de carácter extraño y borrado

Posteriormente, se realiza una reducción de la dimensionalidad, al decidir utilizar el valor resultado de las diferencias de las fechas tanto esperadas y las reales, para tener la cantidad de días que se demoró en realizar un pago y el valor de la diferencias entre las fechas inicio de cobertura y el último día del mes, que indica la cantidad de días de vinculación de las empresas, en vez de los campos tipo fecha separados, ya que de esta manera proporciona mayor información y son más sencillos de operar

Adicionalmente, se decide eliminar el campo estado_empresa_arp_id dado que como sus valores son indicadores actuales de “MORA” o “EN COBERTURA”, por lo que dichas descripciones no corresponden a la verdad del momento en el cual se extrajo la información, lo que puede generar sesgo en los valores de salida del modelo. Como se muestra en la imagen 5.

```
#calculo de las diferencias para las fechas
list(map(lambda x: datetime.datetime.strptime(x,'%Y/%m/%d').strftime('%Y-%m-%d'), df['Fecha_Inicio_Cobertura']))
list(map(lambda x: datetime.datetime.strptime(x,'%Y/%m/%d').strftime('%Y-%m-%d'), df['ultimo_dia_mes']))
list(map(lambda x: datetime.datetime.strptime(x,'%Y/%m/%d').strftime('%Y-%m-%d'), df['Fecha_Limite_Pago']))
list(map(lambda x: datetime.datetime.strptime(x,'%Y/%m/%d').strftime('%Y-%m-%d'), df['fecha_real_pago']))

#se agregan los campos calculados al Dataframe original
df[['Fecha_Inicio_Cobertura','ultimo_dia_mes']] = df[['Fecha_Inicio_Cobertura','ultimo_dia_mes']].apply(pd.to_datetime)
new_col = (df['ultimo_dia_mes'] - df['Fecha_Inicio_Cobertura']).dt.days
df.insert(11, 'cant_dias_cobertura', new_col)

df[['Fecha_Limite_Pago','fecha_real_pago']] = df[['Fecha_Limite_Pago','fecha_real_pago']].apply(pd.to_datetime)
new_col = (df['fecha_real_pago'] - df['Fecha_Limite_Pago']).dt.days
df.insert(13, 'cant_dias_pago', new_col)

#Se eliminan los campos con formato de fecha y el campo estado_empresa_arp_id
df = df.drop('Fecha_Inicio_Cobertura', axis = 1)
df = df.drop('ultimo_dia_mes', axis = 1)
df = df.drop('Fecha_Limite_Pago', axis = 1)
df = df.drop('fecha_real_pago', axis = 1)

df = df.drop('Estado_Empresa_Arp_Id', axis=1)
```

Imagen 4. Primera reducción de dimensionalidad y agregado de campos nuevos para la generación del dataset

El dataframe resultante es persistido en un archivo csv dentro de drive para posteriormente re-usarlo, cuando se necesite correr los modelos posteriormente sin necesidad de hacer nuevamente la limpieza previa.

```
#df.to_csv('/content/drive/Shareddrives/Monografia/dataset_limpio.csv')
df = pd.read_csv('/content/drive/Shareddrives/Monografia/dataset_limpio.csv', delimiter = ",")
df=df.drop('Unnamed: 0', axis=1)
df.head()
```

Imagen 5. persistencia de del dataframe limpio

3.3 DESCRIPTIVA

Explorando los datos, primero se valida cuantos registros únicos por mes tenemos, en este dataset se buscó además que cada identificador de empresa, apareciera solo una vez cada mes, para en caso de abordar el problema como serie de tiempo y no de clasificación, no hubiera que hacer una lógica adicional para volver unico el registro y adicional garantizar que los mismo Ids aparezcan en todo el intervalo de tiempo.

```
Mes_Id
202001    169060
202002    167369
202003    167006
202004    166455
202005    165676
202006    165686
Name: id, dtype: int64
```

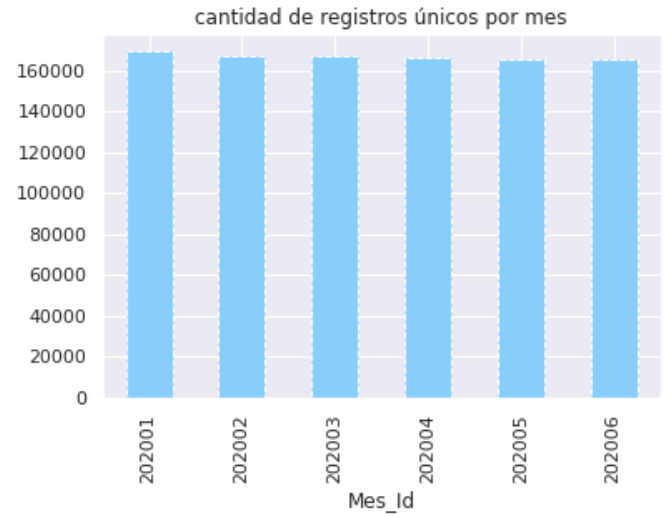


Imagen 5. Distribución de los registros en el intervalo de tiempo

Posteriormente se valida que tan desbalanceado está el dataset

```
en_cartera
0    818511
1    182741
Name: id, dtype: int64
```

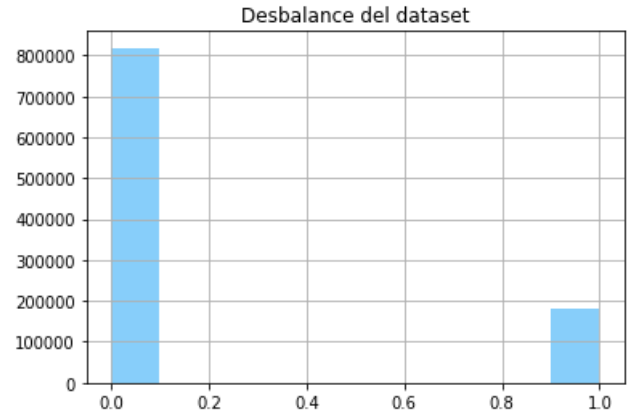


Imagen 6. Desbalance del dataset

Este desbalance es esperado, dado que en caso contrario, la empresa estaría en peligro de quebrarse por falta de pago de los clientes, que no es el caso. Incluso se espera que el desbalance sea mayor en un intervalo de tiempo más amplio.

A continuación, se busca observar si las variables que cambian en el tiempo y dado el nivel de importancia o la cantidad de información que pueda agregar la variable, determinar si es un problema de clasificación o de series de tiempo

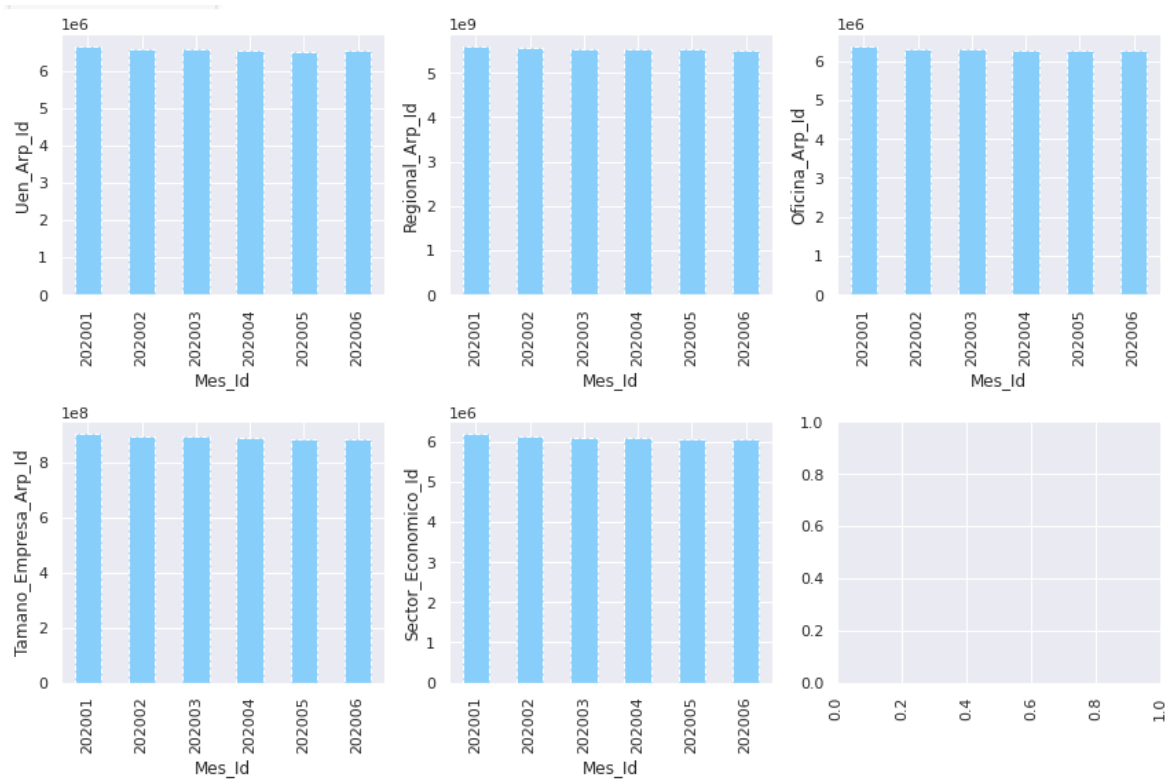
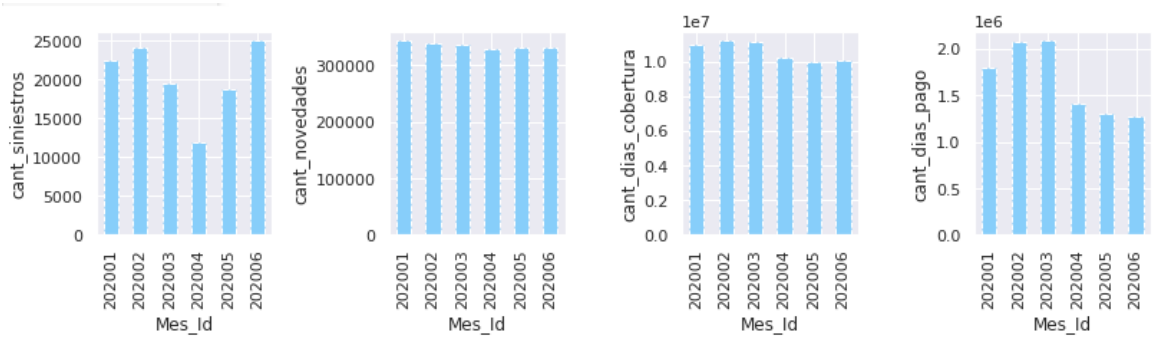
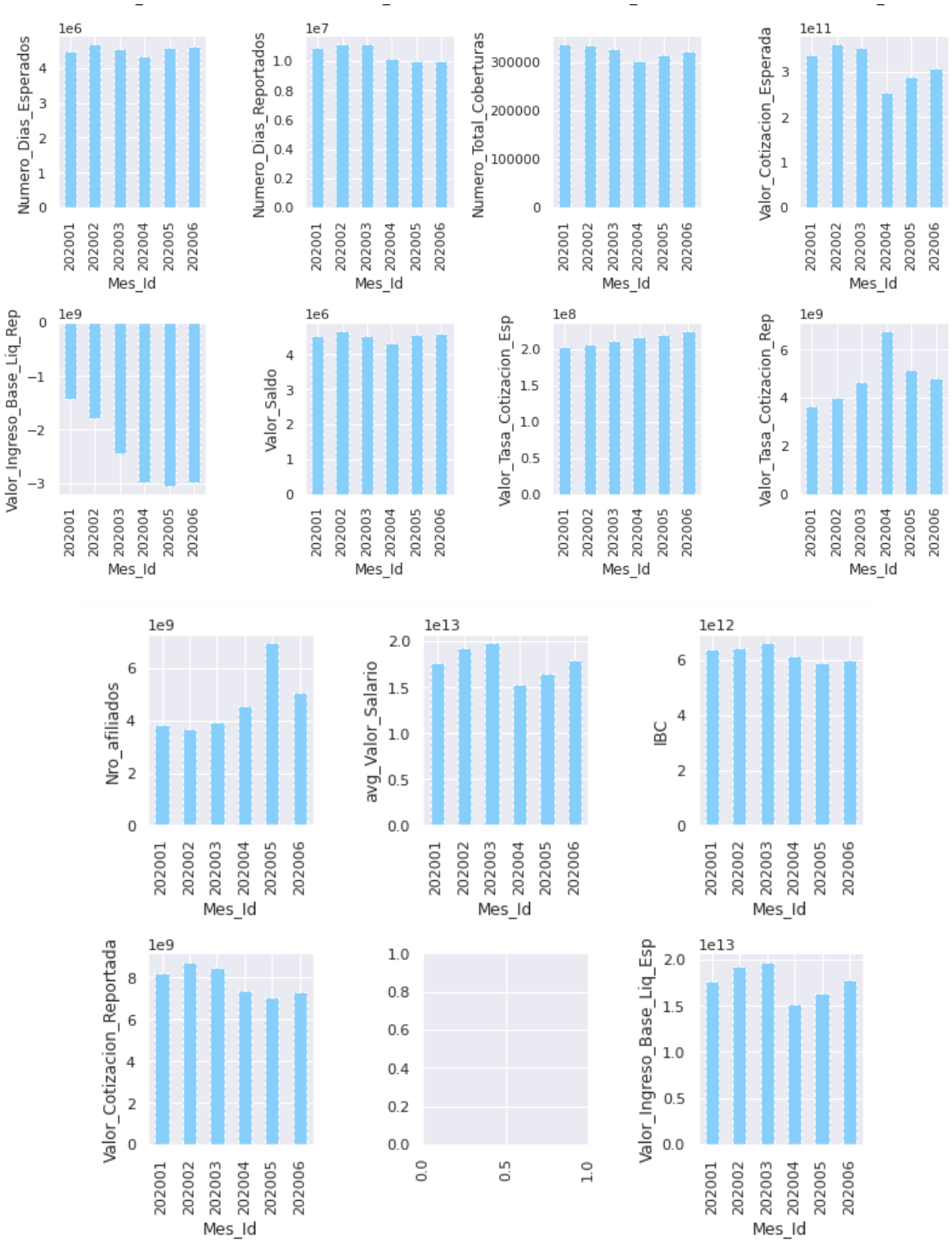


Imagen 7. Variables categóricas respecto al tiempo (mes_id)

Como se espera no hay variaciones perceptibles en las variables categóricas, dado que las características de clasificación, ubicación y sector económico de una empresa rara vez cambian.

En las siguientes cluster de imágenes, se observa el cambio en el tiempo de las variables numéricas




Numero_Dias_Esperados
1e6
Numero_Dias_Reportados
1e7
Numero_Total_Coberturas
Valor_Cotizacion_Esperada
1e11
Valor_Ingreso_Base_Liq_Rep
1e9
Valor_Saldo
1e6
Valor_Tasa_Cotizacion_Esp
1e8
Valor_Tasa_Cotizacion_Rep
1e9
Nro_afiliados
1e9
avg_Valor_Salario
1e13
IBC
1e12
Valor_Cotizacion_Reportada
1e9
Valor_Ingreso_Base_Liq_Esp
1e13
202001
202002
202003
202004
202005
202006
Mes_Id

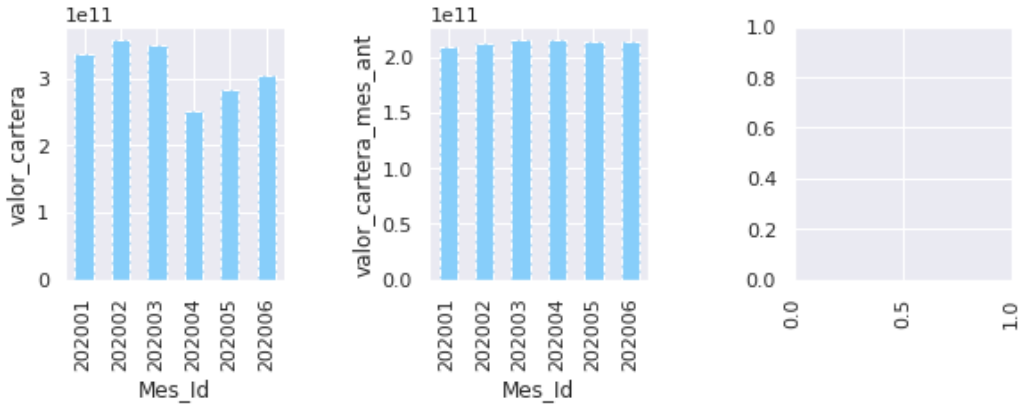


Imagen 8. (cluster de imágenes) Variables numéricas respecto al tiempo (mes_id)

Se observa el cambio en el comportamiento de las variables numéricas, sobretodo en el mes de abril, cuando la pandemia tuvo el pico máximo y luego comenzó la primera reactivación económica.

Además de los cambios atribuidos a los efectos de la pandemia, los datos se comportan, con tendencia, ya sea creciente o decreciente y no muestran cambios en el tiempo, por lo que se trabaja como un problema de clasificación.

4. PROCESO DE ANALÍTICA

4.1 PIPELINE PRINCIPAL

A continuación, en la Fig. X, se presenta un esquema general de un flujo de trabajo integral para cualquier proyecto de analítica y ciencia de datos. Un flujo de este tipo siempre parte de una pregunta de negocio que, para este caso de estudio, consiste en: ¿Qué empresas afiliadas a la ARL tendrán mayor probabilidad de quedar en estado de mora en el mes inmediatamente posterior a la consulta? La Unidad Estratégica de Negocio de la ARL, encargada de la gestión de la cartera, ha construido unas hipótesis de negocio que serán sometidas a validación una vez se implemente el proyecto, durante la etapa de Feedback o retroalimentación. Entre las hipótesis técnicas de las que se tiene conocimiento están:

- Las empresas con una cantidad alta de siniestros tienen mayor probabilidad de quedar en mora en el mes posterior a la consulta.
- La variable 'Estado_Empresa_Arp_Id', cuyos posibles valores son 'EN COBERTURA' o 'EN MORA', es homólogo a la variable a predecir: 'en_cartera '.

- Las variables 'valor_cartera_mes_ant' y 'valor_cartera'están altamente correlacionadas con la variable a predecir: en_cartera; esto pudo comprobarse técnicamente ya que dichas variables sólo son diferentes de cero cuando la empresa está en mora (en_cartera = 1).

La ingestión de los datos se hizo desde el Data Warehouse de la empresa aseguradora de riesgos laborales en estudio; para construir el dataset, se tomaron tres bases diferentes de datos de clientes, las cuales fueron preprocesadas y unificadas en SQL hasta obtener el conjunto descrito en el numeral 3.1. La base de datos resultante fue guardada en un repositorio compartido de Google Drive, con permisos de uso restringido sólo para las autoras de la presente monografía, respetando el acuerdo de confidencialidad de la información que debió firmarse previamente con la ARL. Durante las fases de preparación y análisis de los datos, se aplicaron técnicas de preprocesamiento y análisis descriptivo que serán descritas con mayor detalle en la sección siguiente. Posteriormente, durante el modelado, se construyeron tres modelos base de referencia (o baseline), se evaluaron diferentes métricas de desempeño de los anteriores modelos y se analizaron los resultados obtenidos mediante la matriz de confusión, todo esto con el fin de determinar si se hace necesario o no ajustar los hiperparámetros de los modelos probados, entrenar nuevos modelos o agregar variables externas al sistema que permitan robustecerlo.

Una vez obtenido el mejor modelo o prototipo durante la etapa anterior, éste se someterá a evaluación a través de la predicción de uno o varios registros, conocidos y no conocidos para el sistema; estos registros, al pasar por el prototipo, entregarán una predicción o etiqueta con uno de los dos siguientes valores: 0 (NO MORA) o 1 (EN MORA). Finalmente, dicho modelo será desplegado como servicio para los usuarios del área de Gestión de Cartera en la nube de Microsoft Azure, etapa que se conoce como Integración. Los resultados obtenidos podrán ser validados o confrontados con los usuarios en relación a la pregunta de negocio con la que inició el flujo de los datos de este proyecto, durante la última etapa de F*eedback* o Retroalimentación.

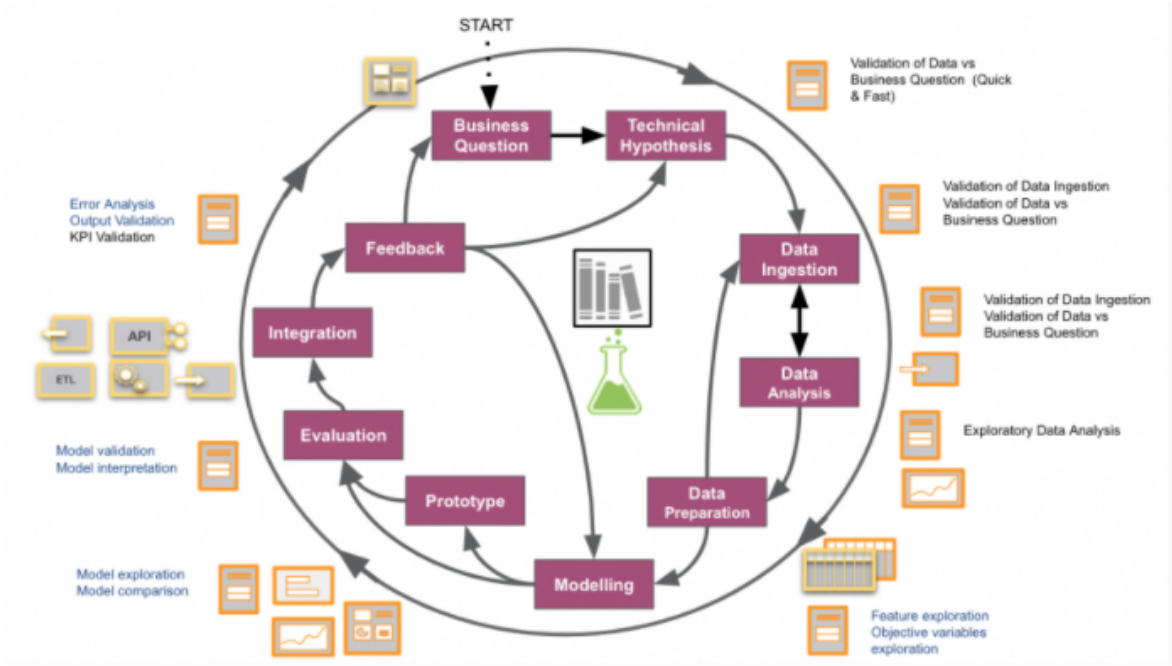


Imagen 9. Esquema general del flujo de los datos del proyecto.
Tomado de: https://www.bbvaaifactory.com/es/accelerating-data-science-workflows/

4.2 PREPROCESAMIENTO

Como se enunció previamente, el dataset otorgado por la compañía proviene de su *DataWarehouse*, los campos fueron seleccionados de mano con una persona experta en el negocio de la ARL y donde el procesamiento que se realizó antes de entregar los datos fue la anonimización de la información, con alguna técnica de cifrado y garantizar la unicidad de los registros respecto al campo Id, es decir garantizar que el mismo Id aparezca solo una vez por mes en todo el intervalo de tiempo, en este caso del 1-ene 2020 al 30-jun 2020.

4.2.1. Preprocesamiento para la primera iteración:

Durante esta primera etapa se aplicaron las siguientes alternativas de preprocesamiento sobre los datos:

- Se verificó la existencia de valores nulos o registros duplicados, con las funciones isnull() y duplicated(), respectivamente. No se encontraron valores nulos ni registros duplicados en el dataset original.

- Al observar la existencia del caracter especial '?' en la columna 'fecha_real_pago', se identificaron y eliminaron los registros que contenían dicho caracter en esta columna; se

realizó la misma verificación para las demás columnas tipo fecha. Se decidió no usar técnicas de amputación para este caso ya que, según recomendaciones del negocio, no deben hacerse suposiciones respecto a la fecha real de pago para evitar introducir sesgos en el sistema. Se identificaron 56.545 registros con esta característica; después de eliminarlos, el nuevo dataset pasó a tener 1.001.252 filas y 31 columnas.

- Se realiza una reducción de la dimensionalidad, al decidir utilizar el valor de la diferencia de las fechas esperada y real para obtener la cantidad de días que el afiliado se demoró en realizar un pago, así como la diferencia entre la fecha de inicio de la cobertura con el último día del mes para obtener la cantidad de días de vinculación de los afiliados, en lugar de considerar los respectivos campos de tipo fecha. El tamaño del nuevo dataset es de 1.001.252 filas por 29 columnas.

- Se decide eliminar el campo 'estado_empresa_arp_id' dado que, como sus valores son indicadores de los estados EN MORA o EN COBERTURA, puede generar sesgo en los valores de salida del modelo. Después de eliminar dicha variable, el nuevo dataset conserva el mismo número de filas, pero ahora contiene 28 columnas.

Una vez realizado el preprocesamiento anterior sobre los datos, el nuevo dataset obtenido se almacena en la carpeta compartida de Google Drive, con el fin de hacer uso de él en etapas posteriores.

4.2.2. Estandarización de los datos: Esta técnica de preprocesamiento se aplica en cada una de las tres iteraciones, una vez se ha realizado la limpieza de los datos indicada en el numeral anterior. Con el fin de expresar todos los valores numéricos de las variables predictoras en una misma escala, se consideraron los siguientes cuatro métodos de escalamiento, y se observó si los modelos son sensibles al tipo de escalamiento que fue aplicado sobre las variables: Standard Scaler, MinMax Scaler, Robust Scaler y Normalizer. Para esto, se construyó en Python la siguiente función, replicable a los demás métodos de escalamiento:

```
def scalerfunct1(dataframe, columnas):
  SS = StandardScaler()
  scaler = SS.fit(dataframe[columnas].values)
  dataframe[columnas] = scaler.transform(dataframe[columnas].values)
  return dataframe
```

Al aplicar Standard Scaler y Robust Scaler se observa que variables como la cantidad de novedades y el número de afiliados, entre otras, toman valores negativos, lo cual podría parecer no lógico a primera vista; sin embargo, este comportamiento obedece a la nueva distribución estadística que siguen las variables después de ajustarlas todas a una misma escala, pero no tienen un significado técnico como tal. Es decir, como estos dos métodos de escalamiento están basados en la media y en cuartiles respectivamente, lo que hacen sobre la distribución de los datos es ajustarlos con

media igual a cero o escalarlos al cuartil 25, lo cual puede producir valores negativos. Adicionalmente, una vez se realice despliegue del modelo de Machine Learning como servicio en la nube, deberán aplicarse las transformaciones inversas respectivas para llevar todos los datos a la escala original.

Lo que se hace a continuación, en la etapa del modelamiento, fue probar los modelos base line con los dos escalados que arrojan valores positivos: MinMax Scaler y Normalizer, y los otros dos métodos que producen negativos: Standard Scaler y Robust Scaler. Si al probar con escalados diferentes, se obtienen resultados diferentes durante el entrenamiento, entonces esto sería evidencia de que los modelos se ven afectados por esas distribuciones estadísticas, y podrá determinarse si alguno de estos escalados está incidiendo o no en obtener un mejor desempeño de los modelos.

Los resultados obtenidos al realizar los anteriores experimentos permiten concluir que los modelos probados no son altamente sensibles al tipo de escalamiento utilizado; sin embargo, según la matriz de confusión, el método que presentó el mejor resultado fue Standar Scaler, por lo que se aplicará este escalamiento de aquí en adelante.

4.2.3. Preprocesamiento para la segunda iteración:

En la segunda iteración se aplicó la técnica de generación de datos sintéticos con la función SMOTE, con el fin de balancear los datos de la clase minoritaria que, para este caso de estudio, corresponde a la clase '1' o 'EN MORA'. Adicionalmente, se realizó una reducción de dimensionalidad sobre el conjunto original de variables aplicando el método de Análisis de Componentes Principales (PCA), con el objetivo de utilizar el menor número de variables no correlacionadas durante el modelamiento, de tal forma que las nuevas variables representen a las antiguas variables de la manera más representativa posible:

```
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier
from imblearn.under_sampling import EditedNearestNeighbours
from imblearn.over_sampling import SMOTE
```

```
pca = PCA(n_components=2)
enn = EditedNearestNeighbours()
smote = SMOTE(random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)
scaler= StandardScaler(with_mean=False)
```

```
from imblearn.pipeline import make_pipeline

model2 = make_pipeline(pca, enn, smote, scaler, knn)
```

Los desempeños de los modelos no mejoraron considerablemente con respecto a los obtenidos de la primera iteración. Dado que este último modelo no aporta una mejora significativa en comparación con los modelos baseline, para evitar la reducción innecesaria de variables predictoras y el aumento infructuoso de la base de datos, se decide descartarse.

4.2.4. Preprocesamiento para la tercera iteración:

Adicionalmente, para la tercera iteración del modelo, se agregaron las siguientes variables macroeconómicas al último dataset, con el fin de ver también reflejado el impacto de la economía local en el comportamiento de pago de las empresas o trabajadores independientes que se encuentran afiliados a la ARL:

- **Índice de Precios al Consumidor (IPC)**: Es una medida de la variación en los precios de los bienes y servicios de consumo más representativos de los hogares colombianos.

Mes	IPC (%)
2020-12	1,61
2020-11	1,49
2020-10	1,75
2020-09	1,97
2020-08	1,88
2020-07	1,97
2020-06	2,19
2020-05	2,85
2020-04	3,51
2020-03	3,86
2020-02	3,72
2020-01	3,62

Tabla A. Valor mensual del IPC durante el año 2020

Tomado de:

https://www.dane.gov.co/index.php/estadisticas-por-tema/precios-y-costos/indice-de-precios-al-consumidor-ipc

- **Índice de Precios al Productor (IPP)**: Es un indicador del cambio de precio neto de venta de los bienes y servicios que conforman la canasta básica familiar, correspondiente al primer canal de comercialización, sin tener en cuenta las modificaciones que hagan los intermediarios al precio de venta del producto antes de llegar al consumidor final.

Mes	IPP
2020-12	124,38
2020-11	124,31
2020-10	124,42
2020-09	123,7
2020-08	123,54
2020-07	122,76
2020-06	122,59
2020-05	122,5
2020-04	122,59
2020-03	123,27
2020-02	122,34
2020-01	122,34

Tabla B. Valor mensual del IPP durante el año 2020

Tomado de: https://www.banrep.gov.co/es/estadisticas/indice-precios-del-productor-ipp

- **Índice de Confianza del Consumidor (ICC)**: Este indicador mide la percepción y las expectativas que tienen los consumidores sobre la economía de un país, y refleja su disposición para realizar gastos en bienes durables e inversiones de largo plazo como la compra de vivienda; por tanto, es un indicador general del comportamiento de la demanda, y de las preferencias y cambios en los hábitos de consumo de las familias colombianas.

Mes	ICC (%)
2020-12	-10,4
2020-11	-13,6
2020-10	-18,6
2020-09	-21,6
2020-08	-25,4
2020-07	-32,7
2020-06	-33,1
2020-05	-34
2020-04	-41,3
2020-03	-23,8
2020-02	-11,2
2020-01	-1,2

Tabla C. Valor mensual del ICC durante el año 2020

Tomado de:
https://www.repository.fedesarrollo.org.co/handle/11445/36/discover?filtertype=dateIssued&filter_relational_operator=equals&filter=%5B2020+TO+2021%5D

- **Tasa de Desempleo (TD)**: Esta tasa mide la proporción del número de personas que se encuentran en busca de empleo, con respecto al número de personas que hacen parte de la población económicamente activa o fuerza laboral.

Mes	TD (%)
2020-12	13,37
2020-11	13,31
2020-10	14,65
2020-09	15,77
2020-08	16,76
2020-07	20,22
2020-06	19,81
2020-05	21,38
2020-04	19,81
2020-03	12,63
2020-02	12,16
2020-01	12,99

Tabla D. Valor mensual de la tasa de desempleo durante el año 2020
Tomado de:
https://www.dane.gov.co/index.php/estadisticas-por-tema/mercado-laboral/empleo-y-desempleo#2020

Estas cuatro variables externas fueron incorporadas al dataset original a través de la función map(), como se presenta a continuación:

```
X['IPC'] = X['Mes_Id'].map(IPC)
X['IPP'] = X['Mes_Id'].map(IPP)
X['ICC'] = X['Mes_Id'].map(ICC)
X['TD'] = X['Mes_Id'].map(TD)
```

Como puede verse en la parte derecha de la imagen 11:

Valor_Saldo	Valor_Tasa_Cotizacion_Esp	Valor_Tasa_Cotizacion_Rep	IPC	IPP	ICC	TD
0.0	0.522	0.522	0.0372	122.34	-0.112	0.1216
-9700.0	1.044	0.522	0.0219	122.59	-0.331	0.1981
0.0	2.436	2.436	0.0285	122.5	-0.34	0.2138
0.0	0.522	0.522	0.0219	122.59	-0.331	0.1981
0.0	0.522	0.522	0.0351	122.59	-0.413	0.1981
...	...	...	...	...	...	...
0.0	2.436	2.436	0.0351	122.59	-0.413	0.1981
-9300.0	1.044	0.522	0.0351	122.59	-0.413	0.1981
-22400.0	1.044	0.522	0.0372	122.34	-0.112	0.1216
0.0	0.522	0.522	0.0372	122.34	-0.112	0.1216
0.0	0.522	0.522	0.0362	122.34	-0.012	0.1299

Imagen 11. Variables externas

4.3 MODELOS

Para el baseline de referencia se emplean los algoritmos de clasificación Random Forest Classifier, Gradient Boosting Classifier y LGBM Classifier, con los hiperparametros que tiene asignados por defecto:

```
clf1 = RandomForestClassifier().fit(X_train, y_train)
clf2 = GradientBoostingClassifier().fit(X_train, y_train)
clf3 = LGBMClassifier().fit(X_train, y_train)
```

Adicionalmente se entrenaron los siguientes modelos, dos de los cuales corresponden a un ajuste manual de los hiperparámetros del modelo de clasificación Random Forest Classifier, con el fin de comparar su desempeño frente al baseline.

```
clf4 = BaggingClassifier().fit(X_train, y_train) # Bagging with Random Undersampling
clf5 = RandomForestClassifier(n_estimators=10, class_weight='balanced').fit(X_train, y_train) # Random Forest with Class Weighting
clf6 = RandomForestClassifier(n_estimators=10, class_weight='balanced_subsample').fit(X_train, y_train) # Random Forest with Bootstrap Class Weighting
clf8 = EasyEnsembleClassifier(n_estimators=10).fit(X_train, y_train) # Easy Ensemble
```

4.4 MÉTRICAS

De la librería Metrics de Sklearn, se importan las siguientes métricas de Machine Learning: accuracy_score, balanced_accuracy_score, f1_score, precision_score y recall_score, así como plot_confusion_matrix, classification_report y multilabel_confusion_matrix.

Con el fin de realizar un análisis comparativo entre estas métricas de desempeño para los diferentes modelos entrenados, se elabora el siguiente bloque de código:

```
labels = ['Random Forest', 'Gradient Boosting', 'LightGBM']
measures = ['acc','ppv','recall','f1','bacc']
Performance = []
Performance.append([accuracy_score(y_test,clf1.predict(X_test)), accuracy_score(y_test,clf2.predict(X_test)), accuracy_score(y_test,clf3.predict(X_test))])
Performance.append([precision_score(y_test,clf1.predict(X_test)), precision_score(y_test,clf2.predict(X_test)), precision_score(y_test,clf3.predict(X_test))])
Performance.append([recall_score(y_test,clf1.predict(X_test)), recall_score(y_test,clf2.predict(X_test)), recall_score(y_test,clf3.predict(X_test))])
Performance.append([f1_score(y_test,clf1.predict(X_test)), f1_score(y_test,clf2.predict(X_test)), f1_score(y_test,clf3.predict(X_test))])
Performance.append([balanced_accuracy_score(y_test,clf1.predict(X_test)), balanced_accuracy_score(y_test,clf2.predict(X_test)), balanced_accuracy_score(y_test,clf3.predict(X_test))])

x = np.arange(len(labels))*2
width = 0.2

fig, ax = plt.subplots()
for i in range(len(measures)):
    ax.bar(x + i*width + 0.3-0.7, Performance[i], width, label=measures[i])

ax.set_ylabel('Scores')
ax.set_title('Scores by measure')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(ncol=1,loc='center right', fontsize='small',bbox_to_anchor=(1.2, 0.5))
fig.tight_layout()
plt.show()
```

La matriz de confusión de cada modelo se calcula reiteradamente de la siguiente manera:

```
disp = plot_confusion_matrix(clf1, X_test, y_test, display_labels=np.unique(y), cmap=plt.cm.Blues, normalize='true')

disp.ax_.set_title('MC normalizada - Random Forest')

print(disp.confusion_matrix)

plt.show()
```

Donde lo que se busca no es solo precisión, sino una correcta clasificación.

5. METODOLOGÍA

5.1 BASELINE

Se escogen dos algoritmos de clasificación Random Forest Classifier, Gradient Boosting Classifier y el algoritmo de regresión LGBM Classifier, con sus hiperparametros por defecto para tener un valor base (baseline), para tener un punto de referencia para poder comparar y mejorar los posteriores modelos.

Dado que es solo la primera corrida, no se tiene en cuenta el desbalanceo de los datos para la partición de los mismos, solo se quiere observar la respuesta de los modelos.

5.2 VALIDACIÓN

Se hace una partición del dataset, donde se deja el 70% de los datos para el entrenamiento y del 30% restante el 25 para test y 0.5 para la validación.

```
[22]  1 X_train, X_test, y_train, y_test = train_test_split(X_n, y, test_size=0.3, random_state=0, stratify=y)
      2 X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.5, random_state=0, stratify=y_test)
```

```
1 # Distribución de las etiquetas - Aplica sólo para Prueba 2 y 4 (Validación)
2 u = ['y_train', 'y_test', 'y_val']
3 v = [y_train.shape[0], y_test.shape[0], y_val.shape[0]]
4 plt.bar(u, v, color=['#1E90FF','#00BFFF','#4682B4'])
5 plt.show()
```

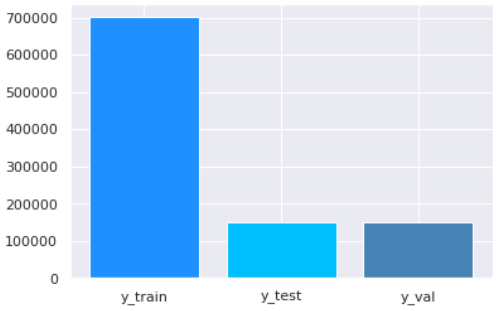


Imagen 12. Partición del dataset en entrenamiento, pruebas y validación

5.3 ITERACIONES y EVOLUCIÓN

Dado que se tiene algunas variables que reflejan temporalidad y también etiqueta de la salida se escogen 2 modelos de clasificación y uno de regresión para evaluarlos.

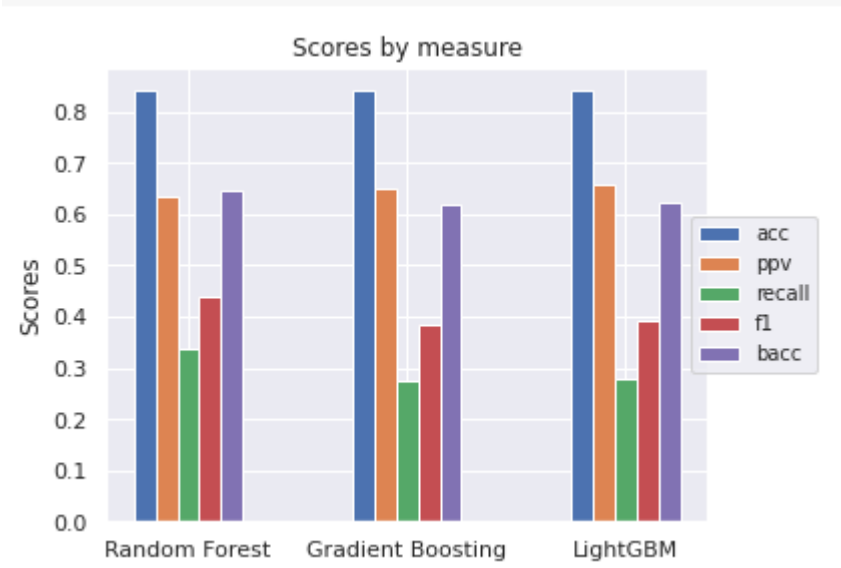
```
1 clf1 = RandomForestClassifier().fit(X_train, y_train)
2 clf2 = GradientBoostingClassifier().fit(X_train, y_train)
3 clf3 = LGBMClassifier().fit(X_train, y_train)
```

```
[25]  1 # Se calcula el score de testeo
      2
      3 print(clf1.score(X_test, y_test))
      4 print(clf2.score(X_test, y_test))
      5 print(clf3.score(X_test, y_test))
```

```
0.843309718486164
0.8403800569952327
0.8419514208858231
```

```
[ ]   1 # Se calcula el score de validación
      2
      3 print(clf1.score(X_val, y_val))
      4 print(clf2.score(X_val, y_val))
      5 print(clf3.score(X_val, y_val))
```

```
0.8417783045249954
0.8388353263909234
0.8401936239912643
```



5.4 HERRAMIENTAS

La herramienta principal fue la extensión de Colab de Google, que nos permite trabajar los notebooks con las diferentes librerías de sklearn y python.
Para la prueba del despliegue del modelo en un entorno que simula el productivo, se hace uso de Visual Studio Code, Docker y los servicios de AzureML.

6. RESULTADOS

6.1 MÉTRICAS

Los resultados numéricos de las métricas para las iteraciones más relevantes fueron los siguientes:

6.1.1. Primera iteración (o baseline):

En el gráfico siguiente se presenta la matriz de confusión de cada uno de los modelos de clasificación descritos en el numeral 4.3. Como puede observarse el algoritmo Easy Ensemble es el que mejor clasifica los datos, dado que se confundió menos a la hora de predecir la clase o etiqueta correcta. Por lo tanto, se escoge éste como el mejor baseline para configurar los siguientes modelos en la segunda iteración, donde se realiza el ajuste de los hiperparámetros.

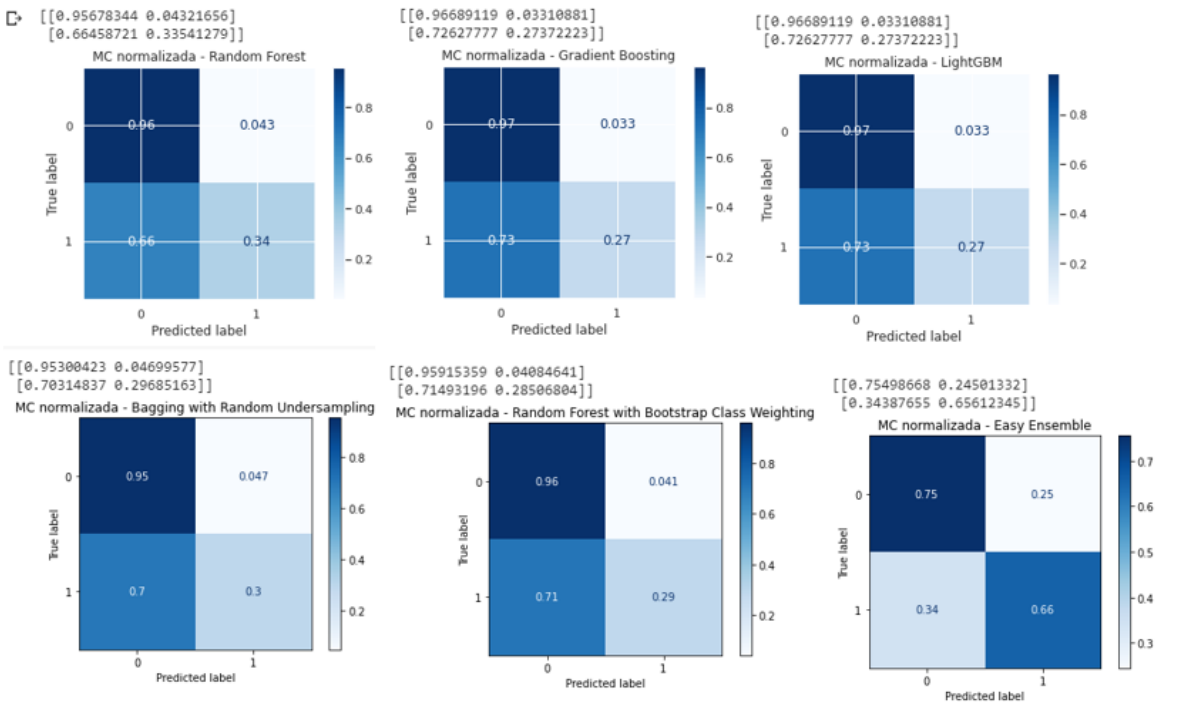


Imagen 13. Resultados de los modelos para el Baseline

6.1.2. Siguientes iteraciones:

Se comienza con la búsqueda de los mejores hiperparametros y agregaciones de datos tanto sintéticos como externos para los diferentes modelos.

En la segunda iteración, se prueban los modelos variando sólo sus hiperparametros, en la tercera iteración se varían los hiperparametros y adicionalmente se agregan datos sintéticos con *SMOTE* y en la tercera iteración se repiten los pasos anteriores, pero se aumenta los campos del dataset agregando las variables externas.
A Continuación se presentan los resultados:

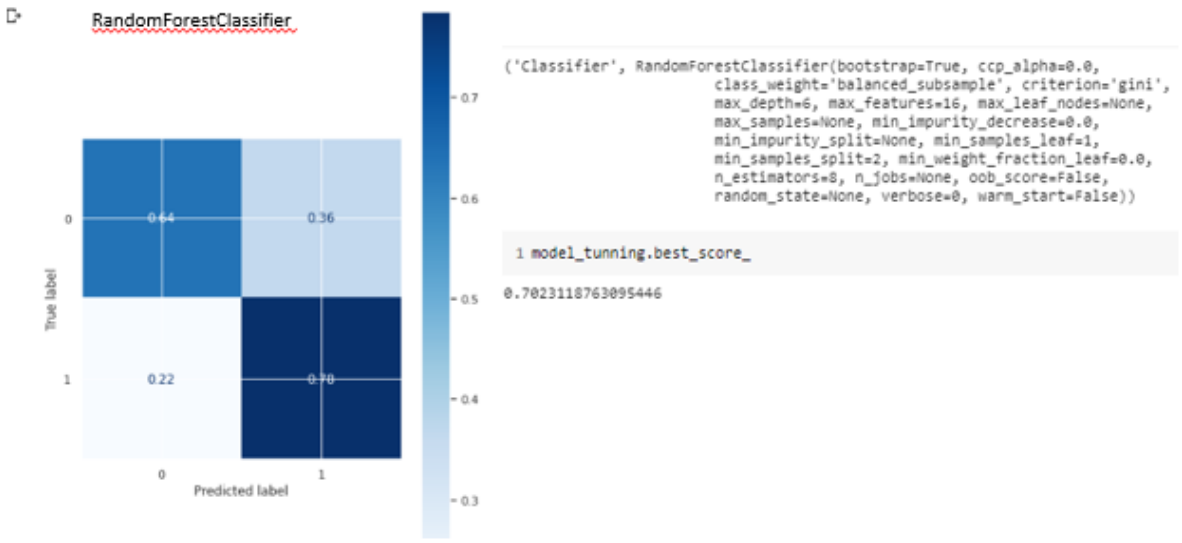


Imagen 14. Mejor resultado para el RandomForestClassificator

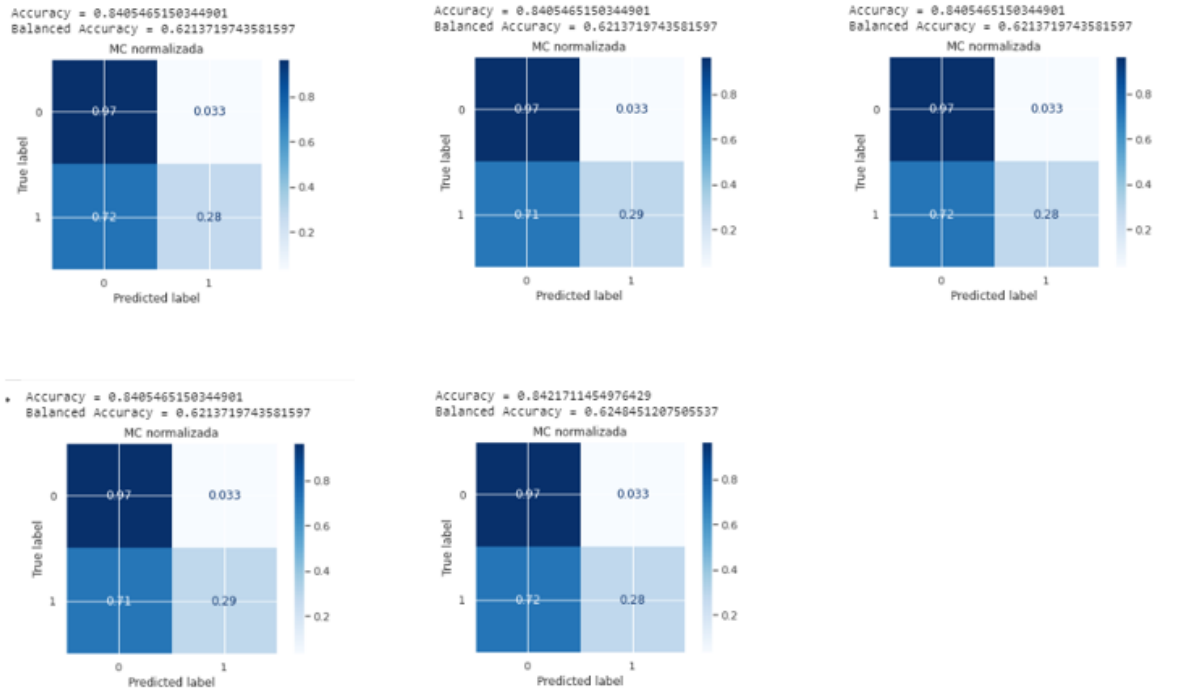
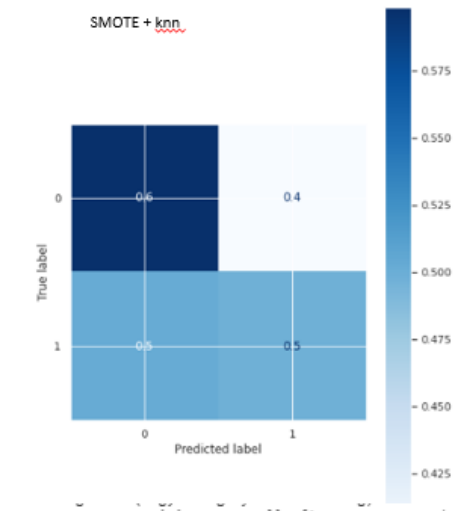
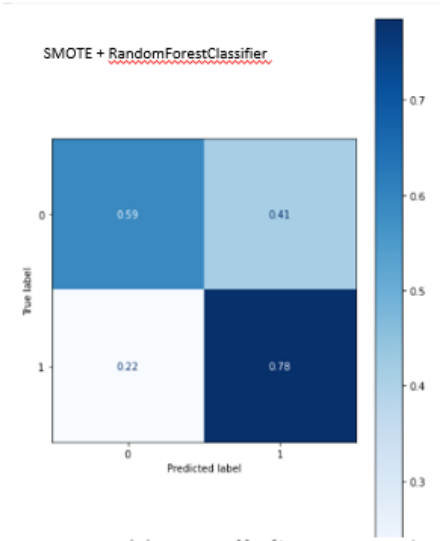


Imagen 15. Mejores resultados para distintos hiperparametros GradientBoostingClassifier



	precision	recall	f1-score	support
0	0.85	0.50	0.63	122777
1	0.21	0.60	0.31	27411
accuracy			0.52	150188
macro avg	0.53	0.55	0.47	150188
weighted avg	0.73	0.52	0.57	150188



	precision	recall	f1-score	support
0	0.90	0.78	0.84	122777
1	0.38	0.59	0.46	27411
accuracy			0.75	150188
macro avg	0.64	0.69	0.65	150188
weighted avg	0.80	0.75	0.77	150188

Imagen 16. Resultados agregando datos sintéticos

6.2 EVALUACIÓN CUALITATIVA

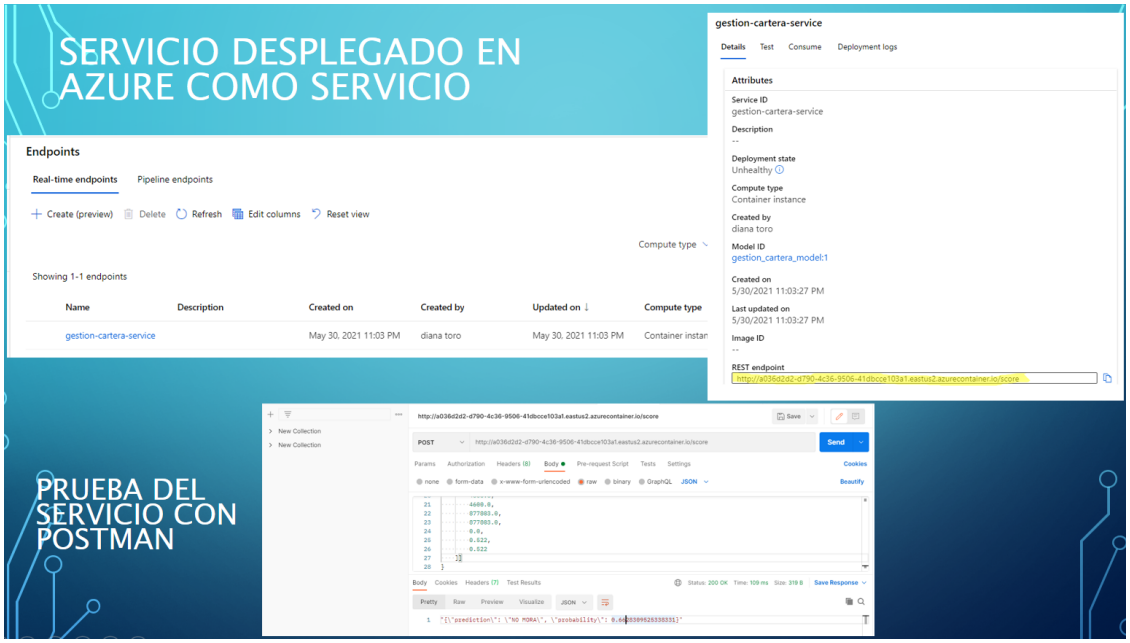
Se observa que los resultados tienden a empeorar cuando se agregan tanto datos sintéticos como datos externos, por lo que el modelo mejor resultado obtenido es el RandomForestClassifier con los siguientes hiperparámetros:

('Classifier', RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
class_weight='balanced_subsample', criterion='gini',
max_depth=6, max_features=16, max_leaf_nodes=None,
max_samples=None, min_impurity_decrease=0.0,
min_impurity_split=None, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
n_estimators=8, n_jobs=None, oob_score=False,
random_state=None, verbose=0, warm_start=False))

y el score obtenido para este modelo es del 70%

6.3 CONSIDERACIONES DE PRODUCCIÓN

La versión para ser desplegada como servicio en la nube de Azure, el tutorial para el despliegue contenerizado junto con el resultado de la prueba del consumo del modelo como servicio en la nube y en ambiente local (con Docker) con Postman, se encuentra en el repositorio: https://github.com/anaidg/clase_cloud_azure_ml.git



7. CONCLUSIONES

Los resultados obtenidos durante las diferentes corridas de los experimentos evidenciaron la importancia de abordar el problema haciendo pequeñas variaciones a los diferentes modelos a ser utilizados y no con todas las consideraciones al tiempo, dado que no siempre más información implica un mejor resultado. También se hizo evidente la importancia de tener un mayor conocimiento del negocio para estar en capacidad de identificar correlaciones entre los datos que no son mostradas técnicamente por un correlograma, y que pueden ocasionar un sobreajuste de los datos arrojando muy buenas pero poco confiables medidas de desempeño para los modelos.

De cara al negocio, contar con un modelo predictivo de clientes morosos al siguiente mes a partir de la consulta, le permitirá a la ARL actuar proactivamente en su día a día, al dotarla de la capacidad de identificar a las empresas que pueden quedar en mora, y ofrecerles con antelación un plan de pagos de alivio financiero según sus características particulares.

Con respecto al resultado del modelo, con los pocos datos que fueron entregados, se logra un buen porcentaje de aciertos, con un intervalo de tiempo más amplio muy probablemente se mejore el score, aunque también el desbalance en las muestras aumentará por lo que se debe

tener en cuenta al momento de entrenar el modelo en producción. Adicionalmente, aunque el mejor modelo encontrado no arrojó un porcentaje muy bajo de falsos positivos y falsos negativos, técnicamente dicho modelo es el que tiene la capacidad predictiva más alta de etiquetar correctamente los verdaderos positivos y verdaderos negativos y, además, dicho modelo es también el que presenta la menor cantidad de falsos positivos. Desde el punto de vista del negocio, es mucho más delicado que un modelo etiquete a una empresa afiliada que se encuentra en mora como si no lo estuviera, ya que esto no permitiría hacer una buena gestión de la cartera; es decir, en este caso, dicha empresa entraría a un estado de cobertura para el mes siguiente a la consulta, cuando debería tener una alerta y convertirse en candidata a acogerse al plan de pagos que la ARL diseñe para proveer alivios financieros. Por el contrario, una empresa que resulte ser un falso negativo sería candidata a optar por un plan financiero que le ofrece la ARL y que realmente no necesita, pero que podría rechazar dada la voluntariedad de acogerse a alguno de estos mecanismos de pago y al sobrecosto que ello le implicaría.