Lo que no se define no se puede medir. Lo que no se mide, no se puede mejorar. Lo que no se mejora, se degrada siempre.

Lord Kelvin.

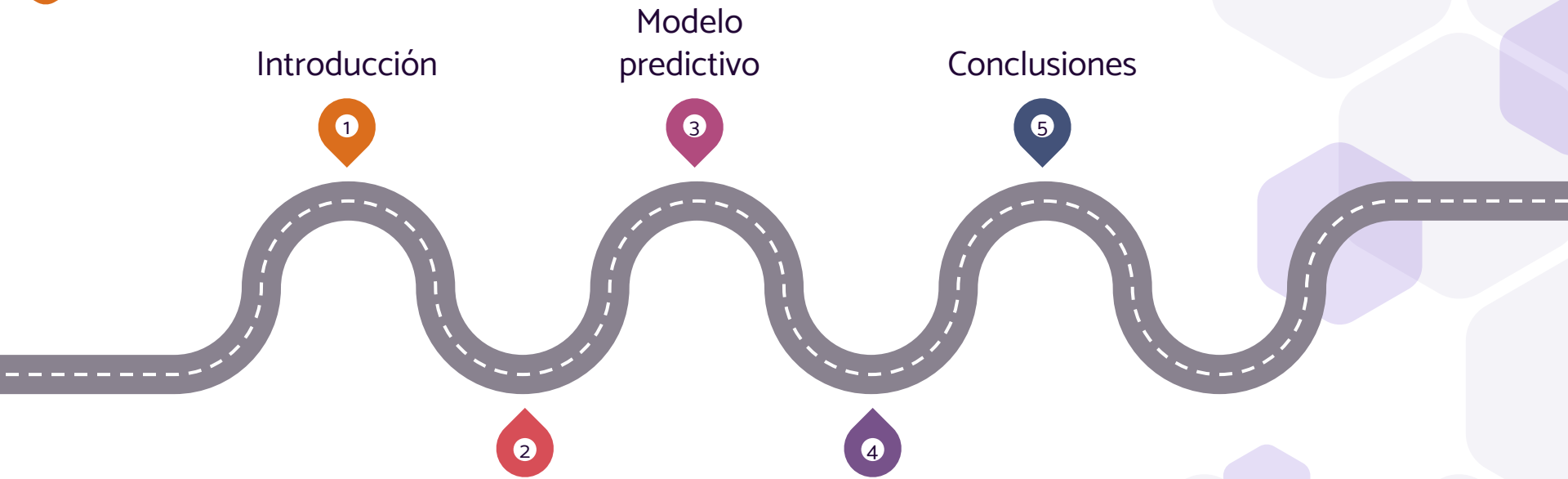
Análisis y predicción de morosidad.

Hola!

Soy Orlando Chacón

@orlandoch

@orlandoch


Introducción
1
Análisis exploratorio
2
Modelo predictivo
3
Sugerencias de implementación.
4
Conclusiones
5

1 Introducción

Contextualización.

Empresa.

Telecomunicaciones:

- Proveedor de internet.
- Redes privadas.
- Integración de soluciones de red.

¿Qué se resolverá?

Análisis exploratorio:

- Concurrencia promedio
  - Hora.
  - Día de la semana.
  - Día del mes.
- Historial de mora.
- Relación Hora-Mora.

Modelo predictivo:

- Probar varios modelos.
- Determinar el mejor modelo según las métricas.
- Sugerencias de implementación..

Sugerencias de implementación.

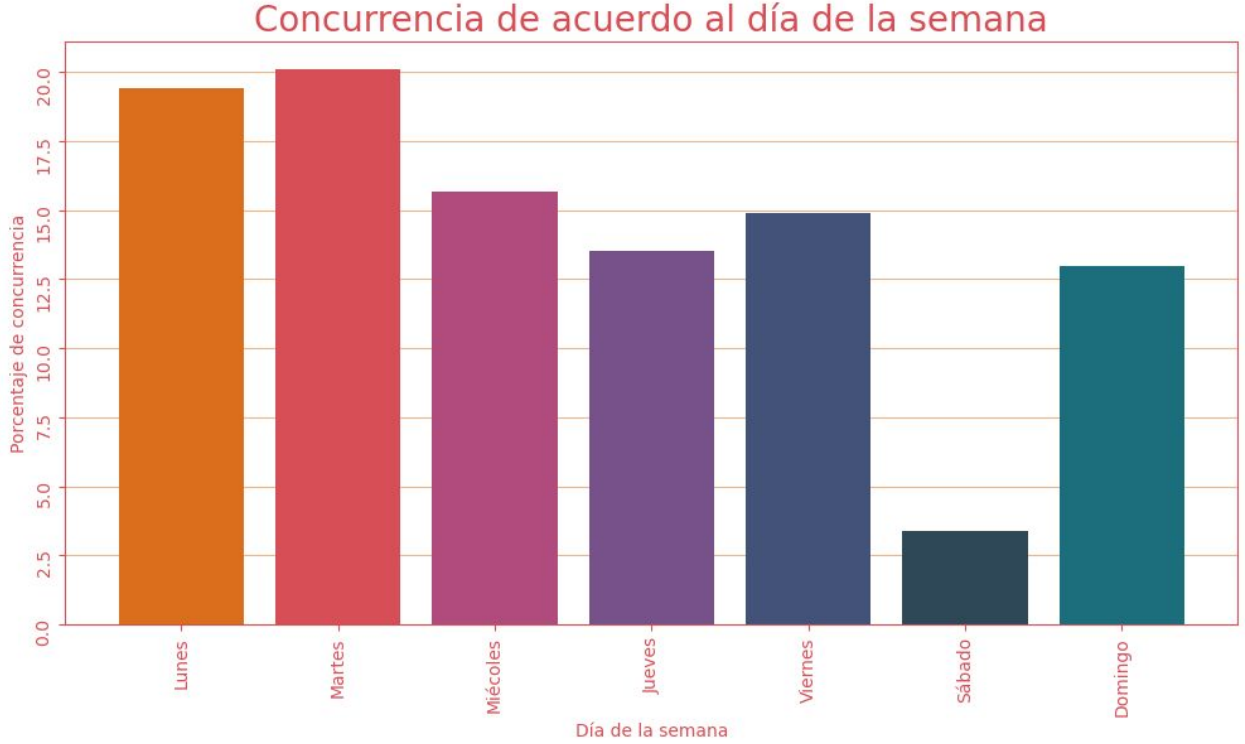
2

Análisis exploratorio.

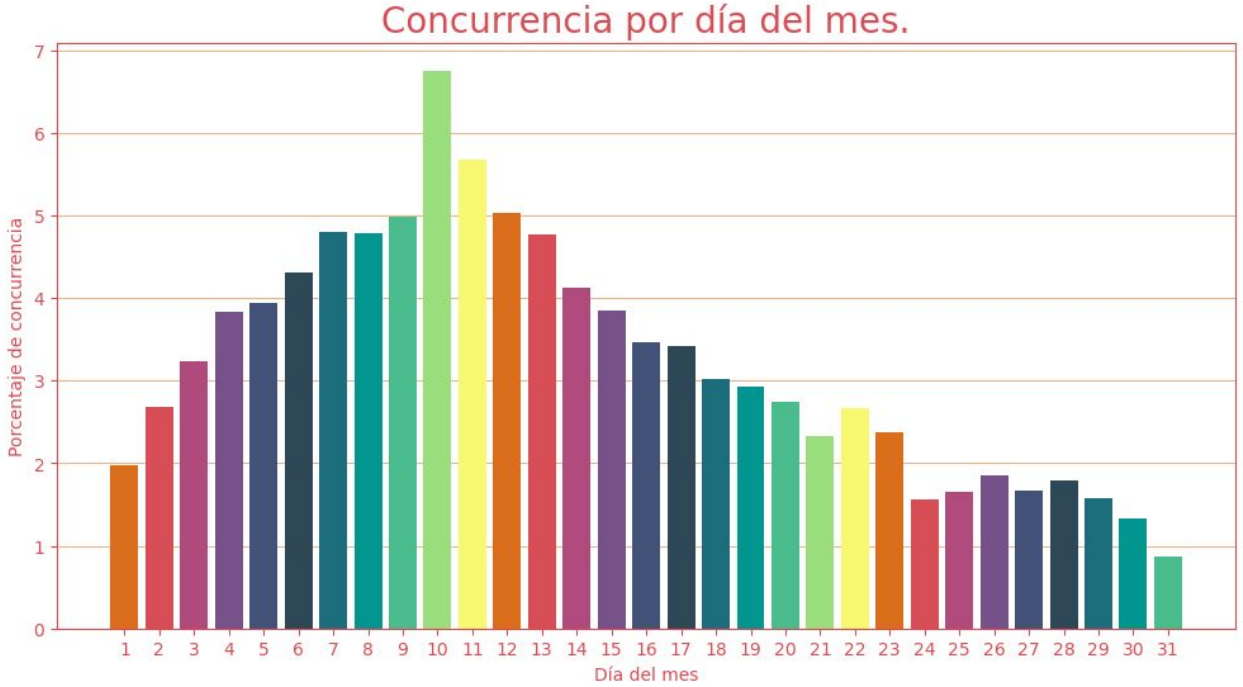
Conocer los datos.

Concurrencia por hora.

Concurrencia día semana.

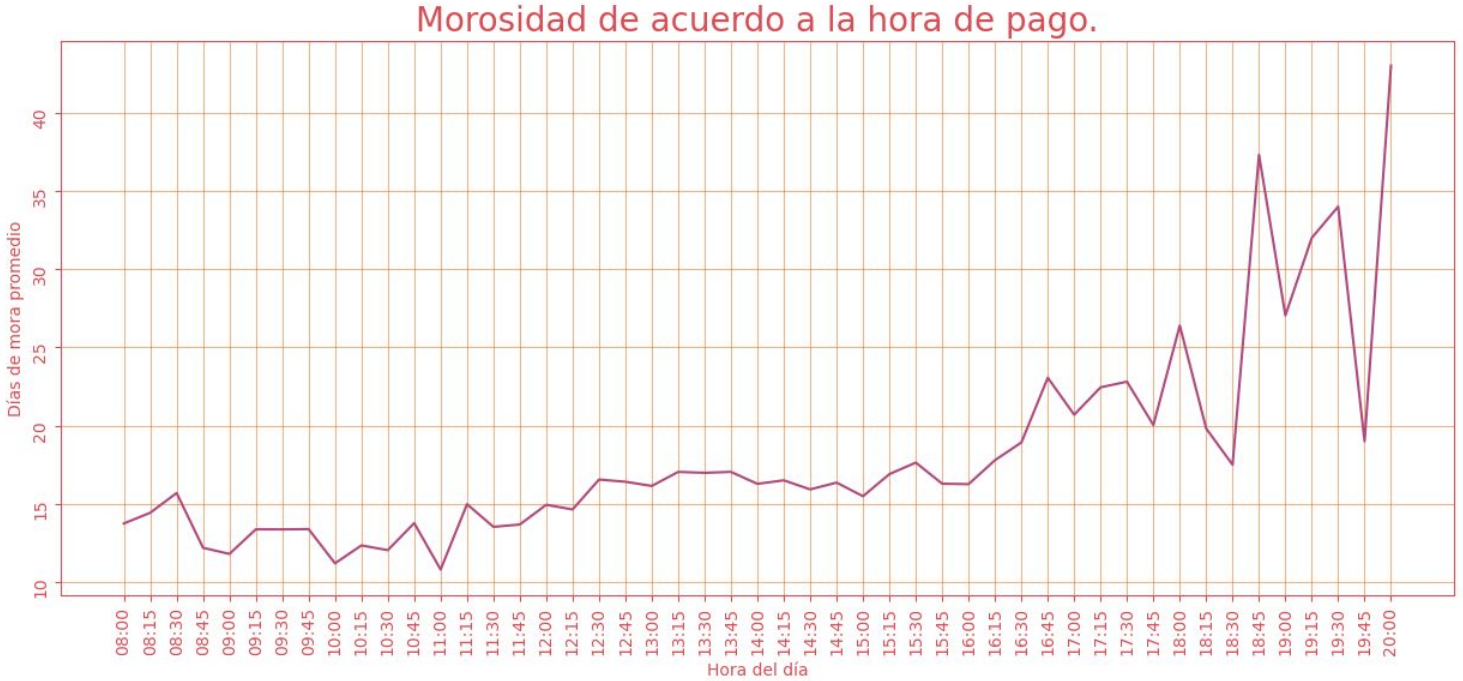


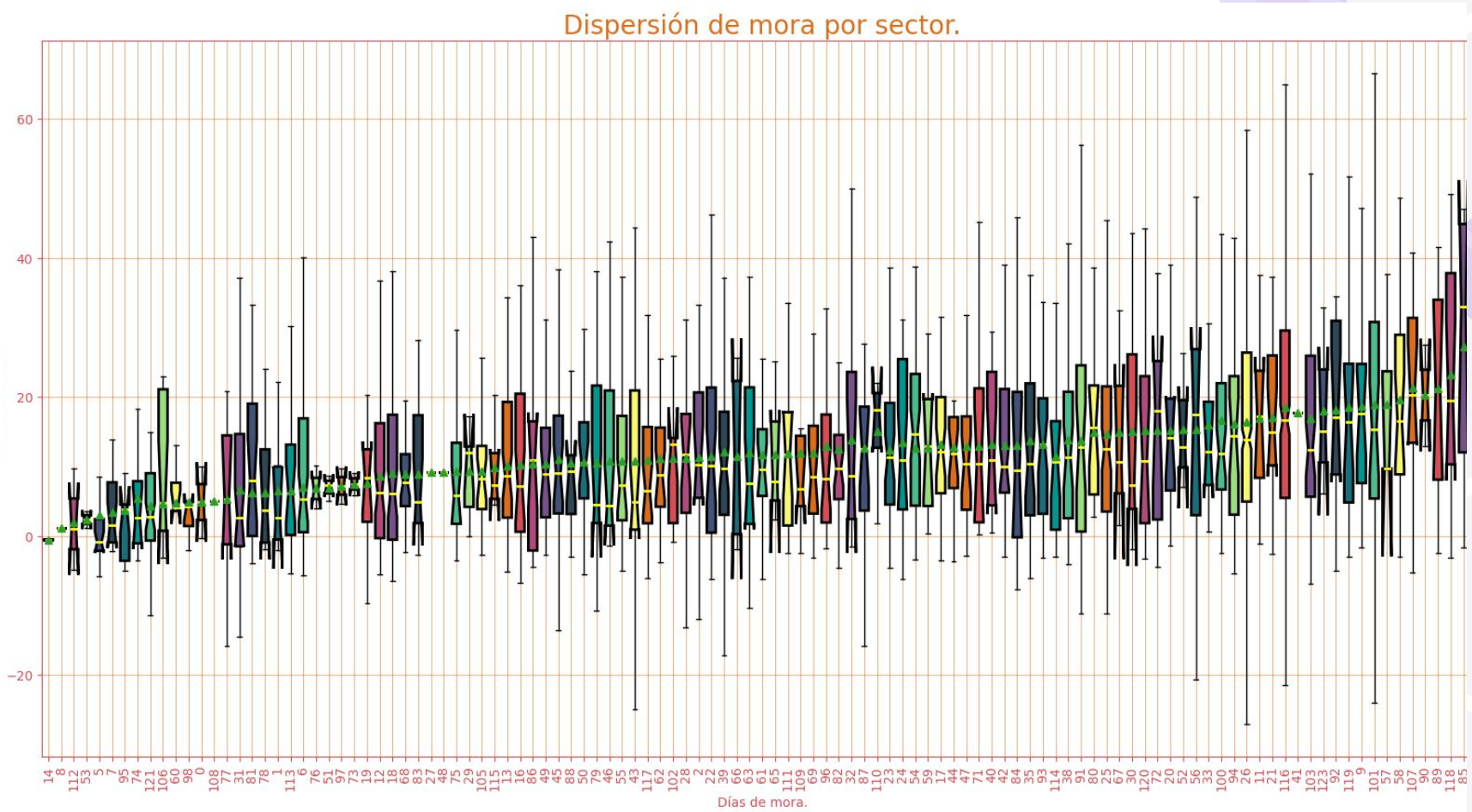
Concurrencia día mes.



Historial de morosidad.

Morosidad por hora de pago.




Dispersión de mora por sector.
60
40
20
0
−20
14 8 112 53 5 7 95 74 121 106 60 98 0 108 77 31 81 78 1 113 6 76 51 97 73 19 12 18 68 83 27 48 75 29 105 115 13 16 86 49 45 88 50 79 46 55 43 117 62 102 28 2 22 39 66 63 61 65 111 109 69 96 82 32 87 110 23 24 54 59 17 44 47 71 40 42 84 35 93 114 38 91 80 25 67 30 120 72 20 52 56 33 100 94 26 11 21 116 41 103 123 92 119 9 101 57 58 107 90 89 118 85
Días de mora.

3

Modelo predictivo.

Determinar mora/no-mora.

Objetivos

- Predecir si un cliente entrará o no en mora.
  - En “tiempo real”.
- Comparar varios modelos y seleccionar el mejor.
  - Varios hiperparámetros.

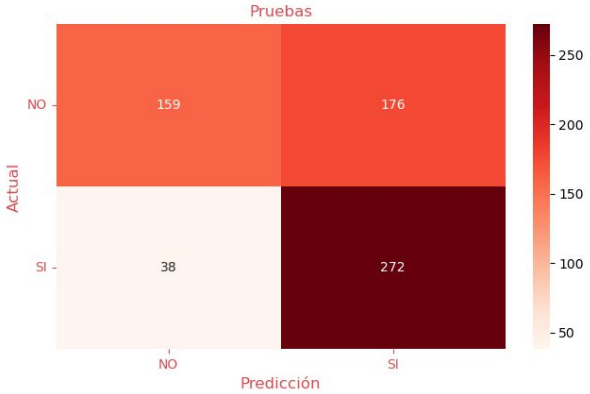
Consideraciones.

- Variables del conjunto de datos original:
  - invoice_id, service_contract_id, neighborhood, contract_days, days_past_due, access_type, contract_plan, lat, lng, emision_date, total_factura, item_price, incident_type, incident_count
- Las variables predictoras deben extraerse en tiempo real.
  - days_past_due_std, neighborhood, contract_plan, incident_count, count, lat, lng
- Se otorgan 10 días de gracia.
  - `0 if x<=10 else 1`

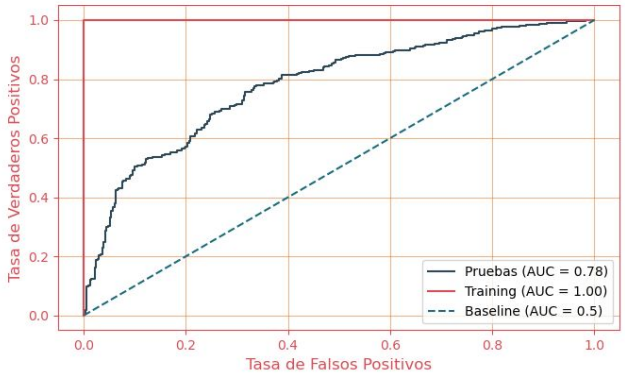
Clasificación: KNN.

5º

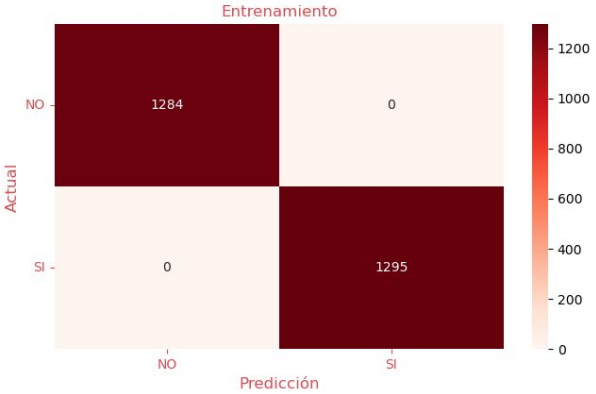
Métricas KNN.(Hiperparámetros ajustados)



Accuracy = 0.668
Precision = 0.607
Recall(Sensibilidad) = 0.877
Specificity = 0.475
F1-score = 0.718



Parámetros utilizados en el modelo:
n_neighbors = 99
weights = distance



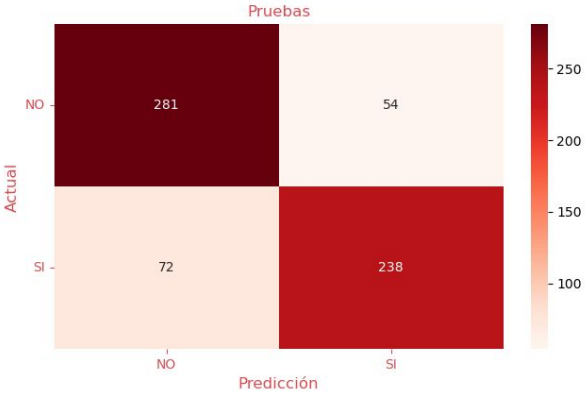
Accuracy = 0.668
Precision = 1.0
Recall(Sensibilidad) = 1.0
Specificity = 1.0
F1-score = 1.0

```
n_neighbors = [i for i in range(1,100)]
weights     = ['uniform','distance']
```

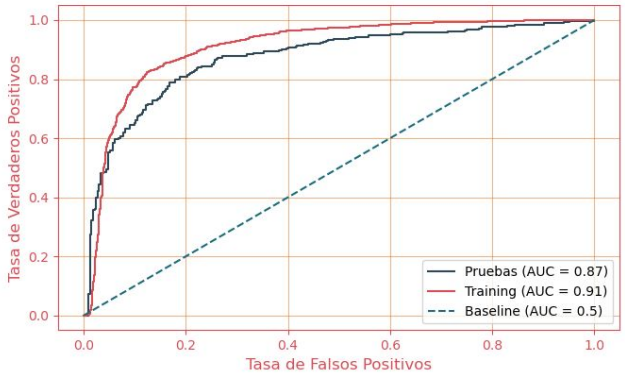
4º

Clasificación: regresión logística

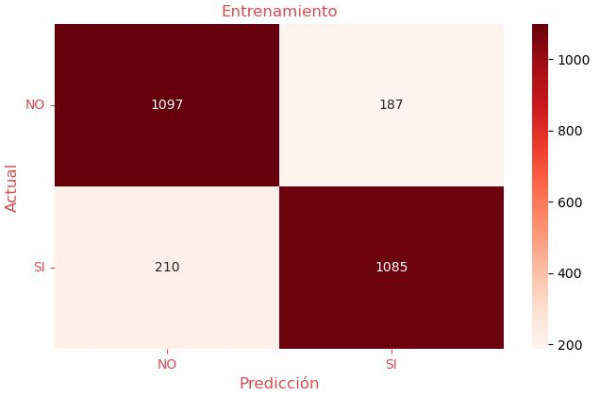
Métricas Regresión Logística(Hiperparámetros ajustados).



Accuracy = 0.805
Precision = 0.815
Recall(Sensibilidad) = 0.768
Specificity = 0.839
F1-score = 0.791



Parámetros utilizados en el modelo:
C = 1.2
max_iter = 10000



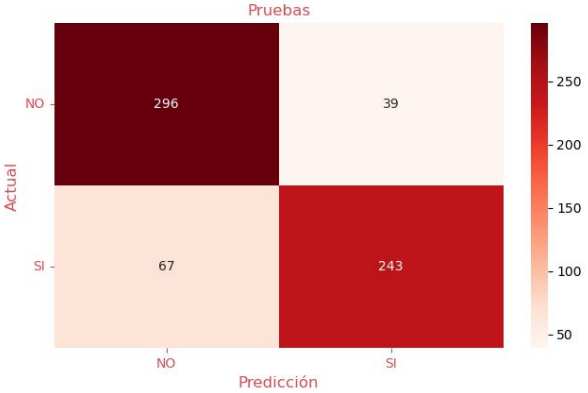
Accuracy = 0.805
Precision = 0.853
Recall(Sensibilidad) = 0.838
Specificity = 0.854
F1-score = 0.845

```
C       = [i*0.3 for i in range(1,15,3)]
penalty = ['l1','l2']
```

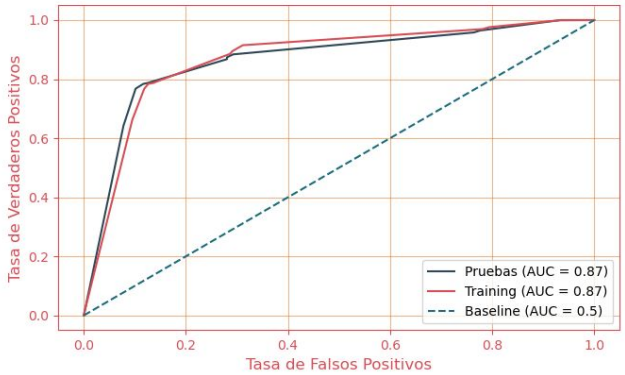
3º

Clasificación: XGBoost.

Métricas XGBoost (Hiperparámetros ajustados).



Accuracy = 0.836
Precision = 0.862
Recall(Sensibilidad) = 0.784
Specificity = 0.884
F1-score = 0.821



Parámetros utilizados en el modelo:
n_estimators = 2
max_depth = 2
subsample = 0.2

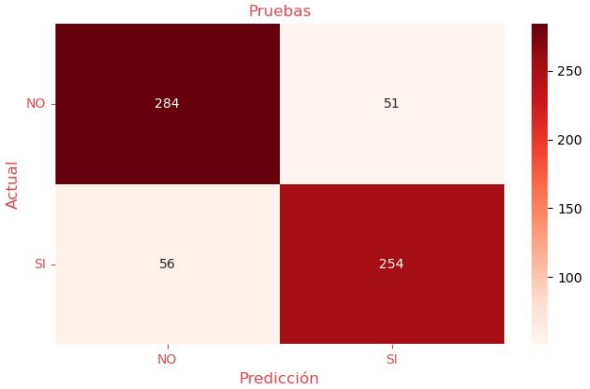
Accuracy = 0.836
Precision = 0.862
Recall(Sensibilidad) = 0.784
Specificity = 0.874
F1-score = 0.821

```
n_estimators = [2,10,100]
max_depth    = [2,4,8]
subsample    = [0.2,0.4,0.6,0.8]
```

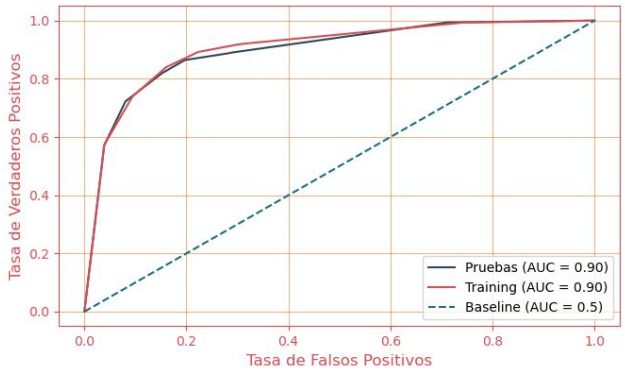
Clasificación: árboles de decisión.

2º

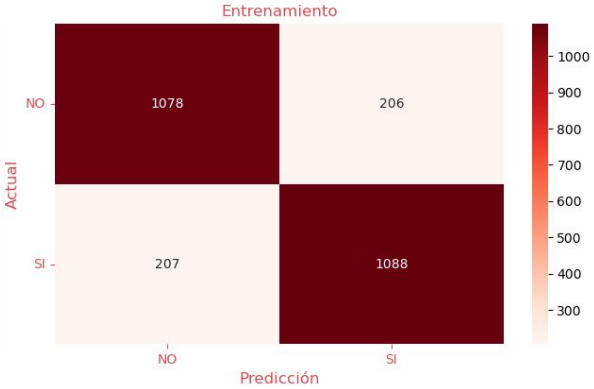
Métricas Árbol de Decisión(Hiperparámetros ajustados).



Accuracy = 0.834
Precision = 0.833
Recall(Sensibilidad) = 0.819
Specificity = 0.848
F1-score = 0.826



Parámetros utilizados en el modelo:
max_depth = 3
criterion = gini

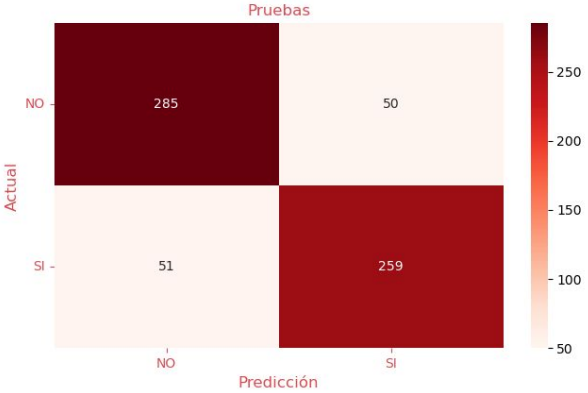


Accuracy = 0.834
Precision = 0.841
Recall(Sensibilidad) = 0.84
Specificity = 0.84
F1-score = 0.84

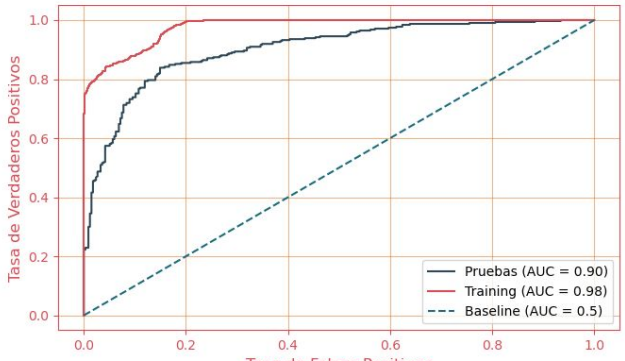
```
max_depth = [i for i in range(1,50)]
criterion = ['gini','entropy']
```

Clasificación: random forest

Métricas Random Forest(Hiperparámetros ajustados).



Accuracy = 0.843
Precision = 0.838
Recall(Sensibilidad) = 0.835
Specificity = 0.851
F1-score = 0.837



Parámetros utilizados en el modelo:
n_estimators = 200
max_depth = 20
bootstrap = True

Accuracy = 0.843
Precision = 0.888
Recall(Sensibilidad) = 0.89
Specificity = 0.887
F1-score = 0.889

```
n_estimators = [200,1000]
max_depth    = [1,20,100]
bootstrap    = [False, True]
```

4

Sugerencias de implementación

Dos alternativas.

Software de cobranzas.

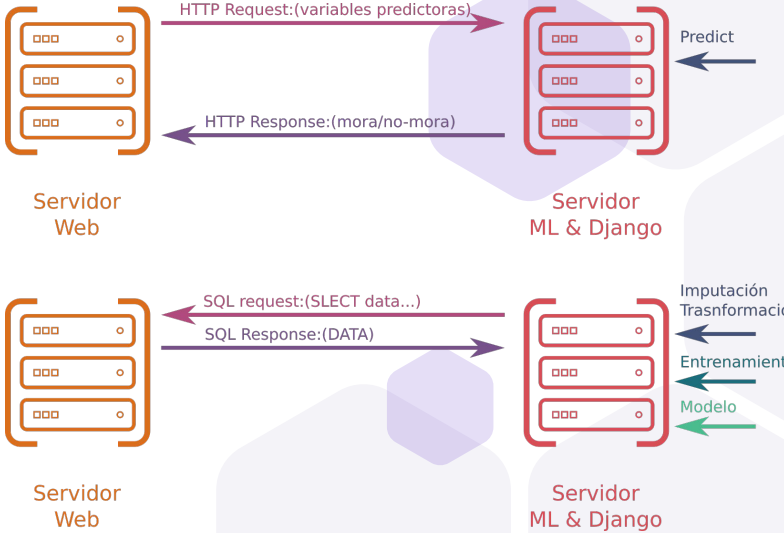
- Entorno Web.
- Framework Laravel (PHP).
- Base de datos MySQL.

1. Dentro del servidor web principal.

- Subir el archivo pickle al servidor web.
- Existen librerías para utilizar los archivos pickle en PHP.
  - Ejecutar el archivo pickle cada vez que se requiera predecir mora/no-mora de un cliente.

2. Servidor dedicado.

- Servidor dedicado (físico o virtual).
- Desarrollar un servicio web con DJango (Python).
  - Recibe variables predictoras.
  - Devuelve mora/no-mora
- Reentrenamiento periódico simple.
  - Obtener datos mediante SQL.
  - Entrenar.
  - Guardar pickle.



5

Conclusiones.

Conclusiones y recomendaciones.

Conclusiones.

- Predicciones aceptables (acc=0.843; F1-Score=0.837).
- Implementación técnicamente factible.
  - Computacionalmente.
  - Adquisición de datos.
- Parte de la morosidad se deriva (Aparentemente) de:
  - Indisponibilidad del cliente dentro del horario de atención.
  - La ubicación geográfica.

Recomendaciones.

- Registrar información adicional.
  - Fecha de aplicación de bloqueo.
  - Fechas de notificación por mora.
    - Contesta / no-contesta.
- Implementar en servidor virtual.
  - La empresa posee servidores.
- Fortificar los canales de pago:
  - Electrónicos.
  - Bancarios.

Medir
Mejorar

Gracias!

¿Preguntas?