



TEC | Tecnológico
de Costa Rica

BIGDATA Programa de Ciencia de los Datos

Proyecto final

Esteban Sáenz Villalobos

Sep. 2021

[\[enlace slides\]](#)

Agenda

Parte 1

Acerca del problema.

Objetivos del proyecto.

Descripción general de
los conjuntos de datos.

Estrategia del proyecto.

Parte 2

Carga,
pre-procesamiento,
ingeniería de
características.

ML (jupyter notebook).

Parte 3

Análisis de resultados
y conclusiones.



Objetivo general

Aplicar técnicas para extracción, transformación, carga de datos realistas de la vida cotidiana y generar predicciones a partir de esos datos depurados.



Objetivo específico

DEPARTURES

FLIGHT

DESTINATION

TIME

STATUS

AB

1350

PARIS

00:25

BOARDING

CD

4521

BERLIN

00:35

GO TO GATE

AF

4370

LONDON

00:40

BOARDING

00:45

BOARDING

00:50

CANCELED

00:50

BOARDING

01:00

BOARDING

01:05

AT 01:45

Predicción de vuelos demorados
para el sector aeronáutico
doméstico de los EEUU.

HA

5160

CHICAGO

Fuentes de datos

Kaggle

Airline Delay and Cancellation Data (2018). La Oficina de Estadísticas de Transporte del Departamento de Transporte de los EE. UU. (DOT) rastrea el desempeño puntual de los vuelos nacionales. La información resumida sobre el número de vuelos puntuales, retrasados, cancelados y desviados se publica en este conjunto de datos de vuelos de 2018.

US Weather Events (2018). Se trata de un conjunto de datos de eventos meteorológicos de todo EEUU que incluye 6,3 millones de eventos y cubre 49 estados. Los datos se recopilan desde enero de 2016 hasta diciembre de 2020 utilizando informes meteorológicos históricos que se recopilaron de 2,071 estaciones meteorológicas en todo el país.

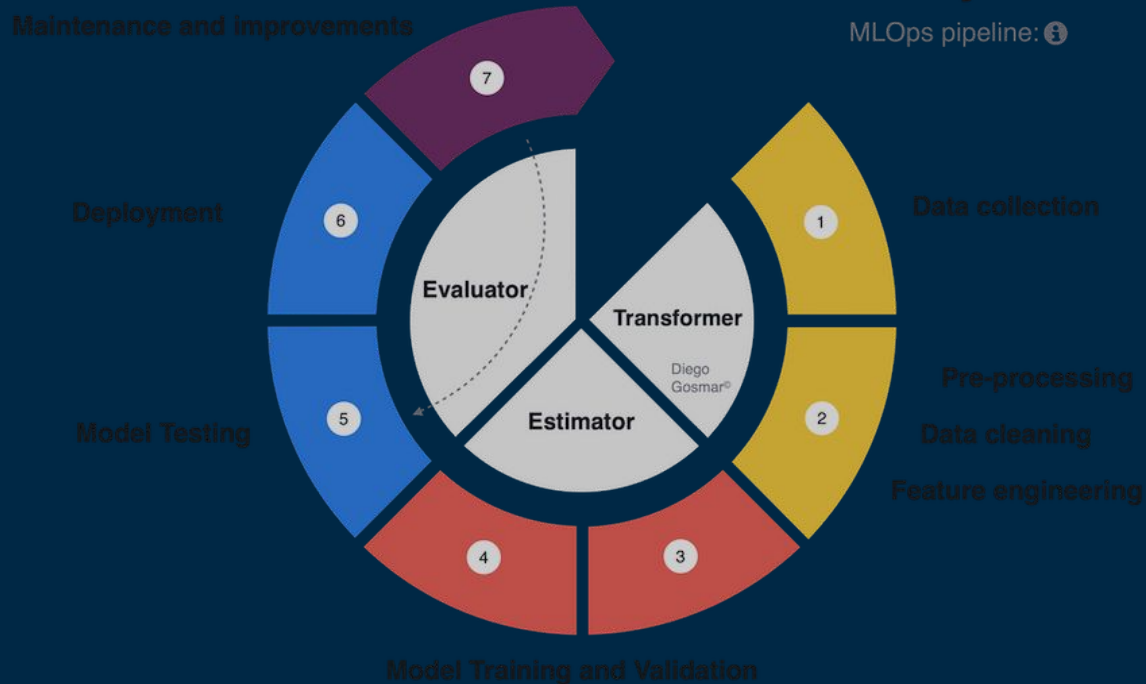
OpenFlights

Open data downloads. La base de datos de aeropuertos de OpenFlights contiene más de 10,000 aeropuertos, estaciones de tren y terminales de ferry en todo el mundo.

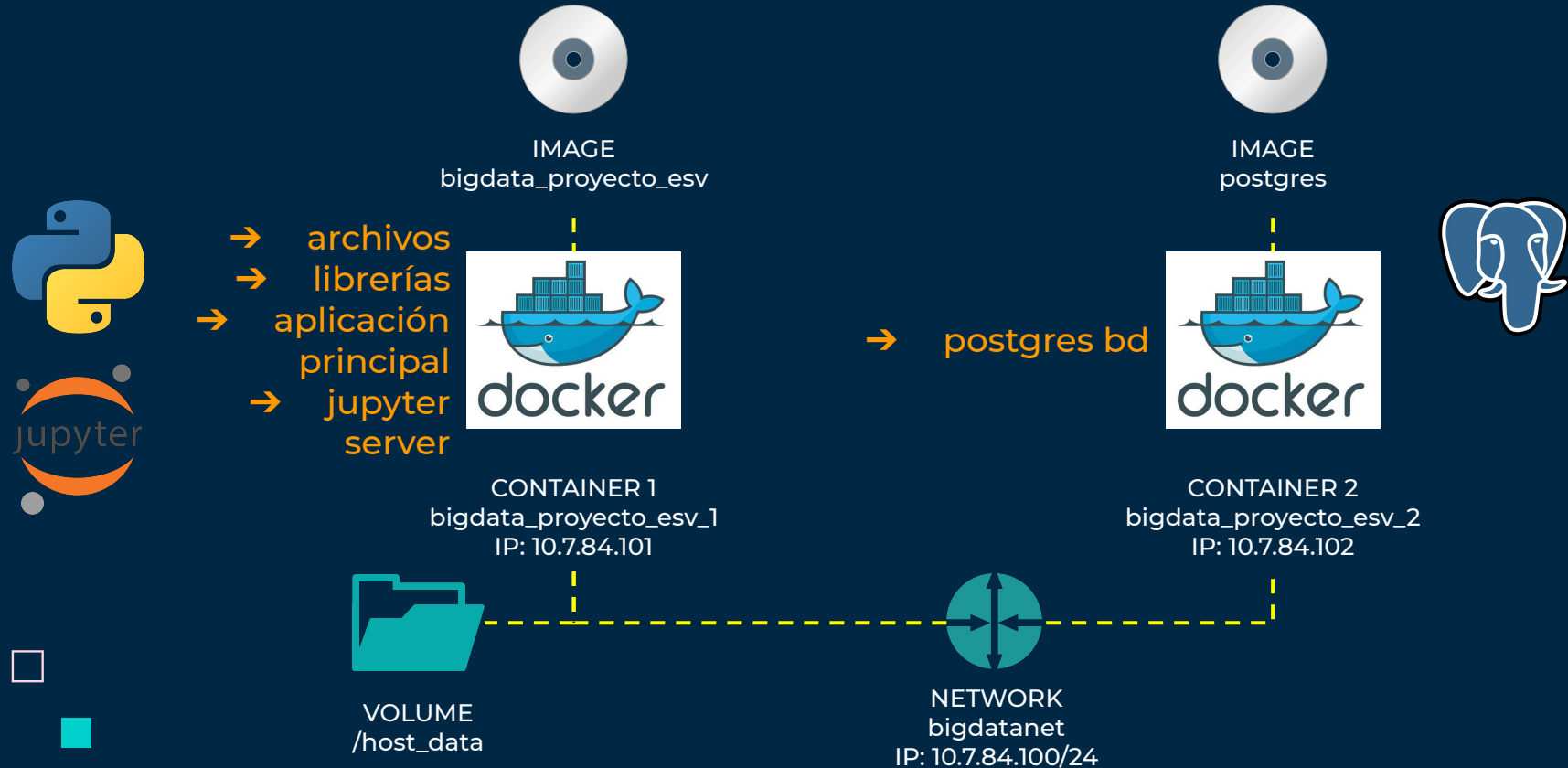
Estrategia

Machine Learning life cycle

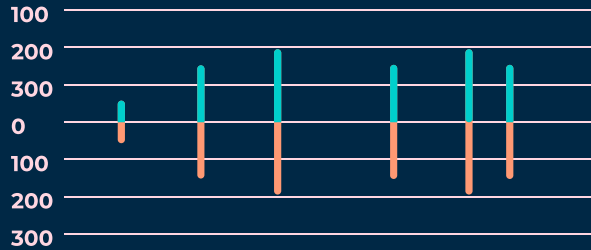
MLOps pipeline: ⓘ



Ambiente de ejecución



Técnicas de preprocesamiento



- Análisis y selección de variables de interés.
- Imputación de valores nulos.
- Indexación y codificación de columnas categóricas.
- Estandarización de los datos.
- Almacenamiento en base de datos.

Almacenamiento

List of relations					
Schema	Name	Type	Owner	Size	Description
public	tb_airports	table	postgres	104 kB	
public	tb_flights	table	postgres	5328 kB	
public	tb_modelolr	table	postgres	0 bytes	
public	tb_modelorf	table	postgres	0 bytes	
public	tb_proyecto	table	postgres	7696 kB	
public	tb_proyectoml	table	postgres	34 MB	
public	tb_weather	table	postgres	33 MB	
(7 rows)					

tb_flights, **tb_airports** y **tb_weather**, conjuntos de datos individuales.

tb_proyecto, conjunto de datos previo al procesamiento.

tb_proyectoml, conjunto de datos preparado (feat. eng.).

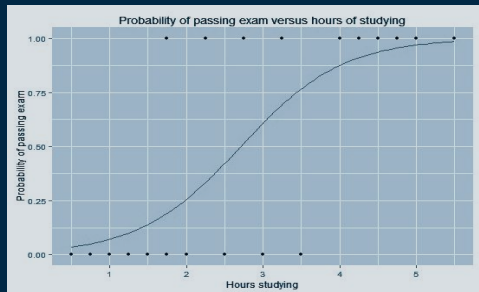
tb_modelolr y **tb_modelorf**, predicciones de cada modelo.



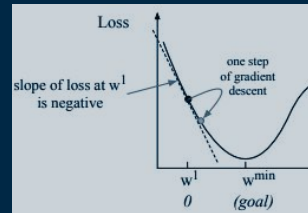
Modelos y entrenamiento

Partición de datos

training    70%
test  30%

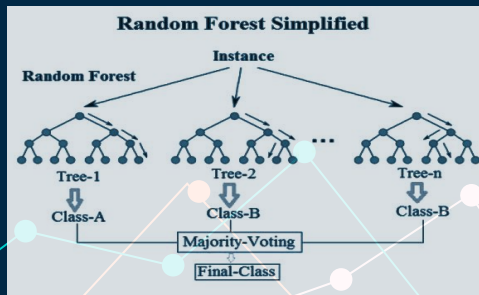


- Regresión logística

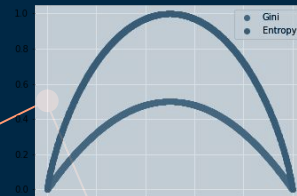


Validación cruzada

k-folds      5



- Bosques aleatorios



Evaluación

- ROC

- PR

Performance metrics associated with Class 1

		Actual Labels	
		1	0
Predicted Labels	1	True Positive	False Positive
	0	False Negative	True Negative

(Is your prediction correct?) (What did you predict)

True Negative

(Your prediction is correct)

(You predicted 0)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{F1 score} = 2 \times \frac{(\text{Prec} \times \text{Rec})}{(\text{Prec} + \text{Rec})}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{False +ve rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

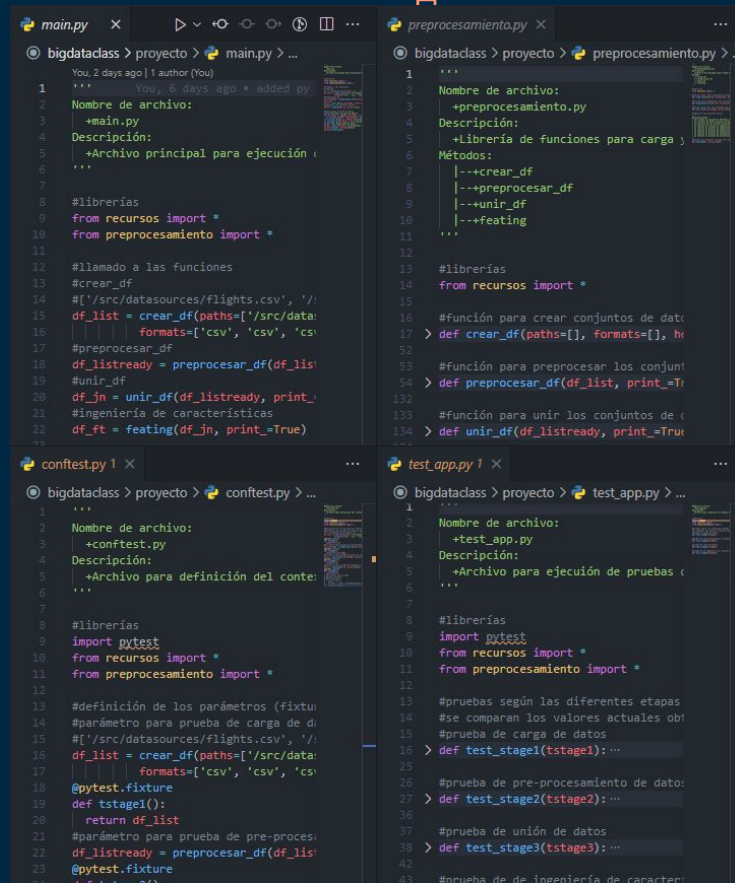
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}}$$

$$\text{Recall, Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

True +ve rate

Ejecución. Parte I

- +recursos.py
- +preprocesamiento.py
- +main.py
- +conftest.py
- +test_app.py



```
main.py
'''
You, 2 days ago | 1 author (You)
'''
You, 6 days ago * added py
Nombre de archivo:
+main.py
Descripción:
+Archivo principal para ejecución
'''
#librerías
from recursos import *
from preprocesamiento import *

#llamado a las funciones
#crear_df
df_list = crear_df(paths=['/src/data:
| | | | formats=['csv', 'csv', 'cs
#preprocesar_df
df_listready = preprocesar_df(df_lis
#unir_df
df_in = unir_df(df_listready, print_
#ingeniería de características
df_ft = feating(df_in, print=True)

preprocesamiento.py
'''
Nombre de archivo:
+preprocesamiento.py
Descripción:
+Librería de funciones para carga
Métodos:
|--+crear_df
|--+preprocesar_df
|--+unir_df
|--+feating
'''
#librerías
from recursos import *

#función para crear conjuntos de dato
> def crear_df(paths=[], formats=[], he
52
#función para preprocesar los conjunt
54 > def preprocesar_df(df_list, print=Tr
132
#función para unir los conjuntos de c
134 > def unir_df(df_listready, print=True

conftest.py
'''
Nombre de archivo:
+conftest.py
Descripción:
+Archivo para definición del conte
'''
#librerías
import pytest
from recursos import *
from preprocesamiento import *

#definición de los parámetros (fixtur
14 #parámetro para prueba de carga de d
15 #['/src/datasources/flights.csv', '/
16 df_list = crear_df(paths=['/src/data:
| | | | formats=['csv', 'csv', 'cs
17 @pytest.fixture
18 def tstage1():
19 return df_list
20 #parámetro para prueba de pre-proces
21 df_listready = preprocesar_df(df_lis
22 @pytest.fixture
23 def tstage2():

test_app.py
'''
Nombre de archivo:
+test_app.py
Descripción:
+Archivo para ejecución de pruebas
'''
#librerías
import pytest
from recursos import *
from preprocesamiento import *

#pruebas según las diferentes etapas
14 #se comparan los valores actuales obf
15 #prueba de carga de datos
> def test_stage1(tstage1): ...
25
#prueba de pre-procesamiento de dato
27 > def test_stage2(tstage2): ...
36
#prueba de unión de datos
38 > def test_stage3(tstage3): ...
42
#prueba de ingeniería de caracteri
```

Conjuntos de datos iniciales

Dataframe 1 (flights.csv)

FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_O
2018-01-01	UA	2141	MSY	DEN	825	819.0	-6.0	9.0	828.0
2018-01-01	UA	2135	DEN	IAD	1735	1740.0	5.0	32.0	1812.0

Dataframe 2 (airports.csv)

_c0	_c1	_c2	_c3	_c4	_c5	_c6
1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	-6.08168983459
2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	-5.20707988739

Dataframe 3 (weather.csv)

EventId	Type	Severity	StartTime(UTC)	EndTime(UTC)	Timezone	AirportCode	LocationLat	Lo
W-967	Snow	Light	2018-01-07 03:54:00	2018-01-07 07:12:00	US/Mountain	K04V	38.0972	-1
W-969	Snow	Light	2018-01-12 13:54:00	2018-01-12 15:12:00	US/Mountain	K04V	38.0972	-1
W-973	Snow	Light	2018-01-21 12:12:00	2018-01-21 16:12:00	US/Mountain	K04V	38.0972	-1
W-974	Snow	Light	2018-01-26 15:12:00	2018-01-26 15:54:00	US/Mountain	K04V	38.0972	-1
W-975	Snow	Light	2018-01-26 20:36:00	2018-01-26 21:12:00	US/Mountain	K04V	38.0972	-1
W-976	Storm	Severe	2018-02-01 17:53:00	2018-02-01 18:53:00	US/Mountain	K04V	38.0972	-1
W-978	Snow	Light	2018-02-06 11:26:00	2018-02-06 11:43:00	US/Mountain	K04V	38.0972	-1
W-979	Snow	Light	2018-02-06 12:34:00	2018-02-06 13:25:00	US/Mountain	K04V	38.0972	-1
W-980	Snow	Light	2018-02-10 21:07:00	2018-02-10 21:41:00	US/Mountain	K04V	38.0972	-1
W-982	Snow	Light	2018-02-10 22:15:00	2018-02-10 22:49:00	US/Mountain	K04V	38.0972	-1

only showing top 10 rows

Ingeniería de características

```
193 176 #ingeniería de características
194 177 '''
195 178 Determinaciones:
196 179 + las clases se encuentran definidas y con un balance aceptable (35/65 aprox.)
197 180 + el conjunto de datos presenta variables tanto categóricas como numéricas
198 181 + las escalas de los valores difieren entre algunas columnas
199 182 + se realiza un proceso de imputación para las variables numéricas
200 183 + se realiza un proceso de imputación para las variables categóricas
201 184 + se realiza un proceso de indexación y codificación para las variables categóricas
202 185 + se realiza un proceso de vectorización para las variables de interés
203 186 + se realiza un proceso de estandarización (se opta por el StandardScaler)
204 187 + se realiza un proceso de extracción de columnas para almacenar en BD
205 188 '''
206
207 list('carrier_'+str(var+1) for var in range(c[0])) +\
208 list('wkday_'+str(var+1) for var in range(c[1])) +\
209 list('month_'+str(var+1) for var in range(c[2])) +\
210 list('morning_'+str(var+1) for var in range(c[3])) +\
211 list('wtyp_'+str(var+1) for var in range(c[4])) +\
212 list('wsev_'+str(var+1) for var in range(c[5])) +\
213 ['label']
214
215 dfcols = spark.createDataFrame(pd.DataFrame(np.array(list(np.append(s.toArray(), 1) for s,l in dfstd.select('scaled','label').collect()))), columns=veccols))
```

Conjuntos de datos preparado

Conjunto de datos preparado

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|carrier|wkday|month|morning|wtyp|wsev|depdel|txout|selap|dist|label|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|WN|1|5|1|
|WN|0|10|0|
|WN|1|5|1|
|WN|1|5|1|
|WN|1|5|1|
|WN|1|5|1|
|WN|1|3|0|
|DL|1|3|0|
|DL|1|3|0|
|DL|1|3|0|
```

Test session starts (platform: linux, Python 3.7.11, pytest 3.6.4, pytest-sugar 0.9.4)
cachedir: .pytest_cache
rootdir: /content, inifile:
plugins: typeguard-2.7.1, sugar-0.9.4

test_app.py::test_stage1 ✓
test_app.py::test_stage2 ✓
test_app.py::test_stage3 ✓
test_app.py::test_stage4 ✓

only showing top 10 rows
Results (11.92s):
4 passed

root

```
-- carrier: string (nullable = true)
-- wkday: integer (nullable = false)
-- month: integer (nullable = true)
-- morning: integer (nullable = false)
-- wtyp: string (nullable = false)
-- wsev: string (nullable = false)
-- depdel: integer (nullable = true)
-- txout: integer (nullable = true)
-- selap: integer (nullable = true)
-- dist: integer (nullable = true)
-- label: integer (nullable = false)
```

Momentos estadísticos

summary	carrier	wkday	month	morning	wtyp	wsev
count	101005	101005	101005	101005	101005	101005
mean	0.0	0.5839512895401218	6.607524379981189	0.41596950646007624	null	null
ctiddev	0.0	0.9790427361967375	2.4454360648107003	0.4928907398152122	null	null

	Clear	Calm
1	Storm	UNK



```
+-----+-----+
|dist|label|
+-----+-----+
|0|0|0|
+-----+-----+
```

```
+-----+-----+
|[0.53425287028633...|1|
|[0.10762112818071...|1|
|[-0.2580632221955...|0|
|[-0.2580632221955...|0|
|[-0.2580632221955...|0|
|[-0.2580632221955...|0|
|[0.27014750612571...|1|
|[-0.2580632221955...|1|
|[-0.0752210470074...|0|
|[-0.3596422084111...|0|
+-----+-----+
```

only showing top 10 rows

Balance de clases objetivo

label	count	%
1	39736	39.34
0	61269	60.66

Visualizaciones

gráfico de parcela

mapa de calor



pandas profiling



Ejecución. Parte II



Partición de datos y parametrización de modelos

Total de observaciones

```
+-----+-----+
|label|count|
+-----+-----+
| 0.0|61269|
| 1.0|39736|
+-----+-----+
```

Entrenamiento

Fracción: 70.23%

```
+-----+-----+
|label|count|
+-----+-----+
| 0.0|42922|
| 1.0|28016|
+-----+-----+
```

Prueba

Fracción: 29.77%

```
+-----+-----+
|label|count|
+-----+-----+
| 0.0|18347|
| 1.0|11720|
+-----+-----+
```

```
23 #parametrización y ajustes del modelo
24 #logistic regression
25 pca_mod = PCA(inputCol='scaled', outputCol='pca')
26 lr_mod = LogisticRegression(featuresCol='pca', labelCol='label')
27 lr_estimador = pipe(stages=[pca_mod,lr_mod])
28 lr_grid = ParamGridBuilder()\
29     .addGrid(pca_mod.k, [10, 25, len(df_4[1].columns[:-1])])\
30     .addGrid(lr_mod.maxIter, [5, 15, 25])\
31     .addGrid(lr_mod.threshold, [.4, .5])\
32     .build()
33 lr_evaluator = BinaryClassificationEvaluator(metricName='areaUnderROC')
34 lr_cv = CrossValidator(estimator=lr_estimador, estimatorParamMaps=lr_grid, evaluator=lr_evaluator, numFolds=5)
35
36 #parametrización y ajustes del modelo
37 #random forest
38 pca_mod = PCA(inputCol='scaled', outputCol='pca')
39 rf_mod = RandomForestClassifier(featuresCol='pca', labelCol='label')
40 rf_estimador = pipe(stages=[pca_mod,rf_mod])
41 rf_grid = ParamGridBuilder()\
42     .addGrid(pca_mod.k, [10, 25, len(df_4[1].columns[:-1])])\
43     .addGrid(rf_mod.numTrees, [25, 50])\
44     .addGrid(rf_mod.impurity, ['entropy', 'gini'])\
45     .build()
46 rf_evaluator = BinaryClassificationEvaluator(metricName='areaUnderROC')
47 rf_cv = CrossValidator(estimator=rf_estimador, estimatorParamMaps=rf_grid, evaluator=rf_evaluator, numFolds=5)
```

Entrenamiento

```
5 #modelado con conjunto de entrenamiento
6 %time lr_cvmodel = lr_cv.fit(df_train)
7 %time rf_cvmodel = rf_cv.fit(df_train)
```

```
CPU times: user 7.59 s, sys: 1.42 s, total: 9.01 s
Wall time: 5min 23s
CPU times: user 7.02 s, sys: 1.19 s, total: 8.21 s
Wall time: 8min 36s
```

```
3 #predicciones con conjunto de prueba
4 %time lr_predic = lr_cvmodel.transform(df_test).collect()
5 %time rf_predic = rf_cvmodel.transform(df_test).collect()
```

```
CPU times: user 2.93 s, sys: 62.4 ms, total: 2.99 s
Wall time: 8.77 s
CPU times: user 2.61 s, sys: 51.8 ms, total: 2.66 s
Wall time: 8.42 s
```

```
1 #resultados de las estimaciones y mejor estimación
2 sf1, sf2, ef = '\n\033[1m\033[106m\033[30m', '\n\033[1m\033[103m\033[30m', '\033[0m'
3 print('Resultados de la evaluación cruzada K-Fold')
4 print(sf2, 'Regresión logística', ef, lr_cvmodel.getEstimator())
5 print(sf1, 'Estimaciones realizadas', ef, ' '.join('{}: {:.2f}'.format(*k) for k in enumerate(lr_cvmodel.avgMetrics)))
6 print(sf1, 'Mejor estimación', ef, lr_cvmodel.getEstimatorParamMaps()[np.argmax(lr_cvmodel.avgMetrics)], '\n')
7 print(sf2, 'Bosques aleatorios', ef, rf_cvmodel.getEstimator())
8 print(sf1, 'Estimaciones realizadas', ef, ' '.join('{}: {:.2f}'.format(*k) for k in enumerate(rf_cvmodel.avgMetrics)))
9 print(sf1, 'Mejor estimación', ef, rf_cvmodel.getEstimatorParamMaps()[np.argmax(rf_cvmodel.avgMetrics)], '\n')
```

Resultados de la evaluación cruzada K-Fold

Regresión logística Pipeline_f68096d5957d

Estimaciones realizadas [0: 0.74] [1: 0.74] [2: 0.74] [3: 0.74] [4: 0.74] [5: 0.74] [6: 0.77] [7: 0.77] [8: 0.77] [9: 0.77] [10: 0.77] [11: 0.77] [12: 0.91]

Mejor estimación {Param(parent='PCA_ed0f35dd9e03', name='k', doc='the number of principal components'): 39, Param(parent='LogisticRegression_cf7082370f42',

Bosques aleatorios Pipeline_bcb6e625c251

Estimaciones realizadas [0: 0.75] [1: 0.76] [2: 0.76] [3: 0.76] [4: 0.77] [5: 0.77] [6: 0.77] [7: 0.77] [8: 0.79] [9: 0.79] [10: 0.79] [11: 0.79]

Mejor estimación {Param(parent='PCA_181a2b075e86', name='k', doc='the number of principal components'): 39, Param(parent='RandomForestClassifier_dc329115db

Evaluación. Conceptos

- **Precision**= $tp / (tp + fp)$, capacidad de no etiquetar como positiva una muestra que es negativa.
- **Recall**= $tp / (tp + fn)$, capacidad de encontrar todas las muestras positivas.
- **F beta**= **ponderación de P y R** (entre 0 y 1), pesado por un factor beta y con un umbral específico. En este contexto los **fp** no son tan costosos como los **fn** y el desequilibrio de clase no cambia la puntuación.
- **ROC**= **compensación entre tpr y fpr**, calcula el **tpr**, el **fpr** y se traza la gráfica para cada umbral. En este contexto importan tanto las clases positivas como negativas, por lo que no se recomienda ante un alto desbalance de clases.

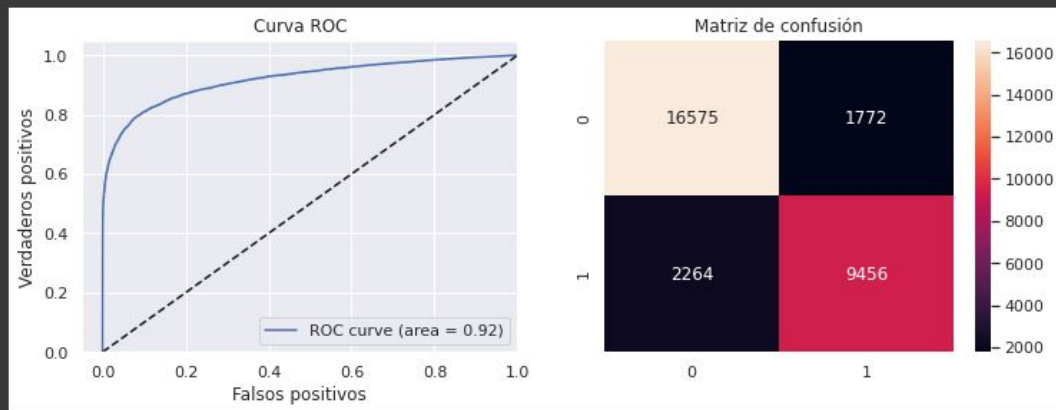
Evaluación. Análisis de resultados.

Regresión logística

Reporte de clasificación

	precision	recall	f1-score	support
0.0	0.88	0.90	0.89	18347
1.0	0.84	0.81	0.82	11720
accuracy			0.87	30067
macro avg	0.86	0.86	0.86	30067
weighted avg	0.87	0.87	0.87	30067

---AUC ROC: 86.10%--- ---AUC PR 77.26%---



86%

ROC regresión logística

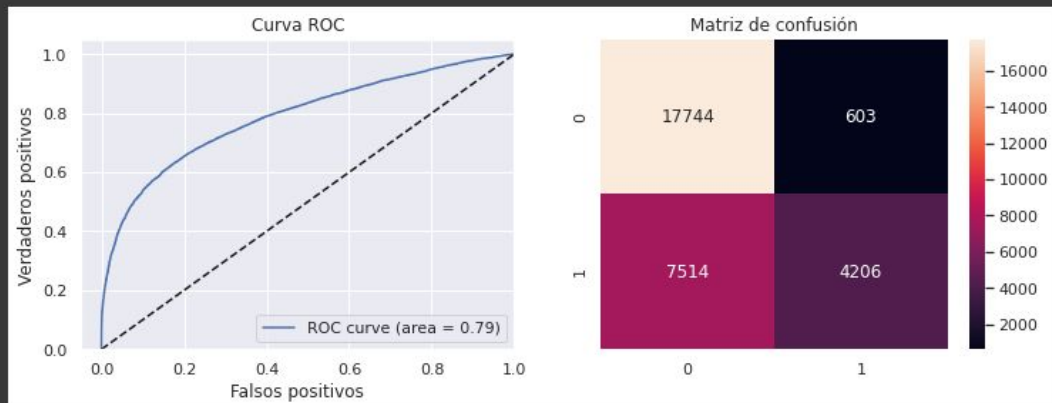
Evaluación. Análisis de resultados.

Bosques aleatorios

Reporte de clasificación

	precision	recall	f1-score	support
0.0	0.70	0.97	0.81	18347
1.0	0.87	0.36	0.51	11720
accuracy			0.73	30067
macro avg	0.79	0.66	0.66	30067
weighted avg	0.77	0.73	0.69	30067

---AUC ROC: 78.86%--- ---AUC PR 34.64%---

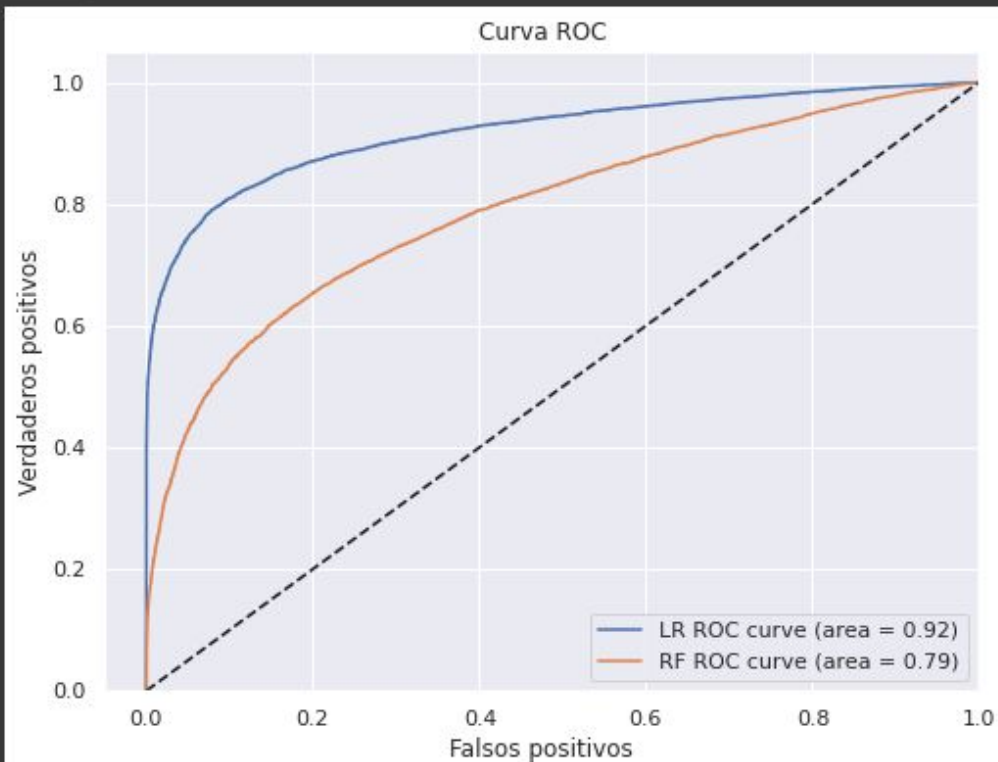


79%

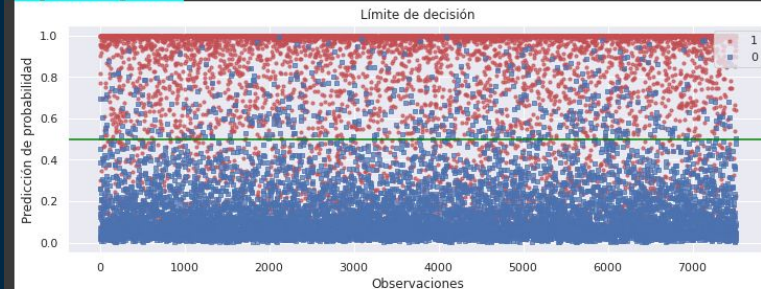
ROC bosques aleatorios



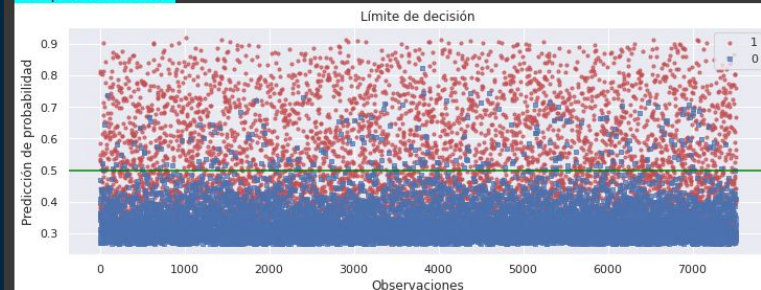
Comparación de resultados



Regresión logística



Bosques aleatorios



Algunas conclusiones acerca del proyecto

- Recordar que spark se basa en una ejecución "lazy".
- Mantener el código ordenado y compacto, documentar todo.
- Aprovechar herramientas como GitHub y Docker.
- Evitar el uso de acciones de pyspark innecesarias o recurrentes.
- Trabajar con muestras de datos pequeñas para realizar pruebas, pero no olvidar de ejecutar el conjunto completo.
- Ajustar los modelos de aprendizaje con diferentes parámetros es importante, pero antes es primordial preparar el conjunto de datos adecuadamente.
- ☐ **Recordar siempre que el aprendizaje automático es solo una herramienta para resolver un problema.**

BIGDATA Programa de Ciencia de los Datos

Proyecto final