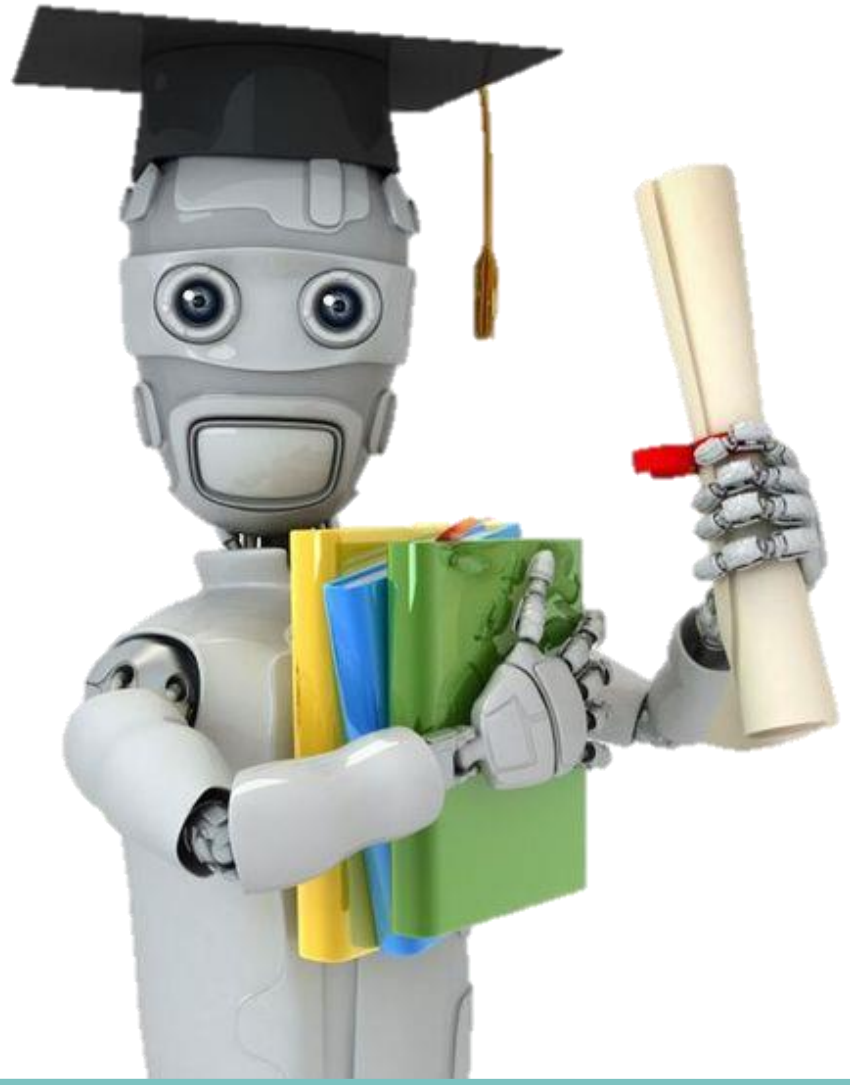
Meetup



Charla 1:

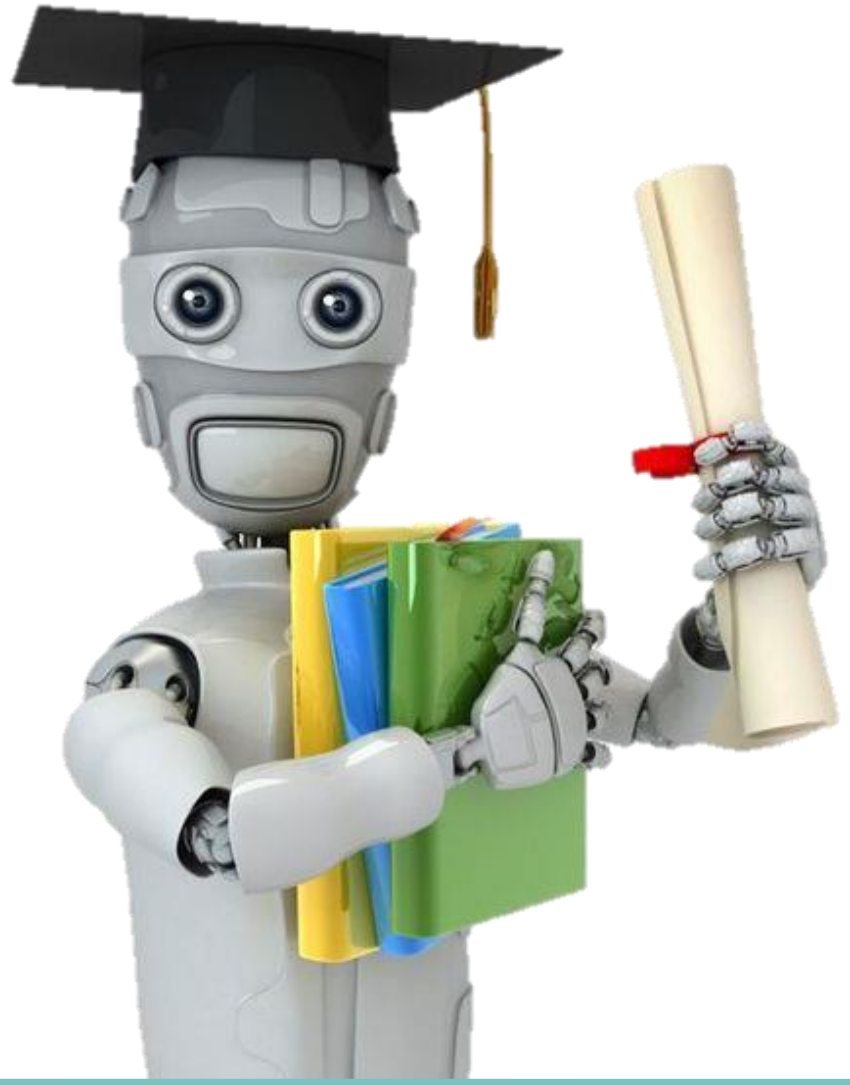
CARACTERÍSTICAS DE SERIES DE TIEMPO PARA UN MODELO DE CLASIFICACIÓN

jupyter R

IBM Data Science Experience

H_2O.ai

Meetup



Charla 1:

CARACTERÍSTICAS DE SERIES DE TIEMPO PARA UN MODELO DE CLASIFICACIÓN

Alex Castaño Ballesteros

github.com/acastanob
linkedin.com/in/acastanob/
acastanob@gmail.com
312 840 88 87

Predicción ¿✗✓?

¿Se Fugará el cliente?

¿Tengo el virus xyz?

¿Comprará otro producto?

¿Eso será Pitufeo?

¿ Es Fraude?

¿Está Lavando Activos?

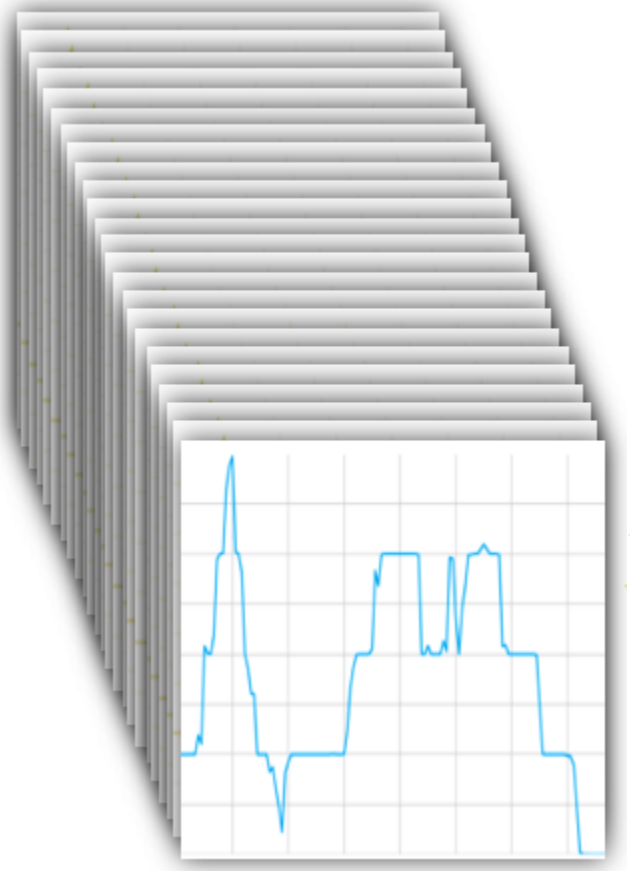
¿Es un cliente potencial?

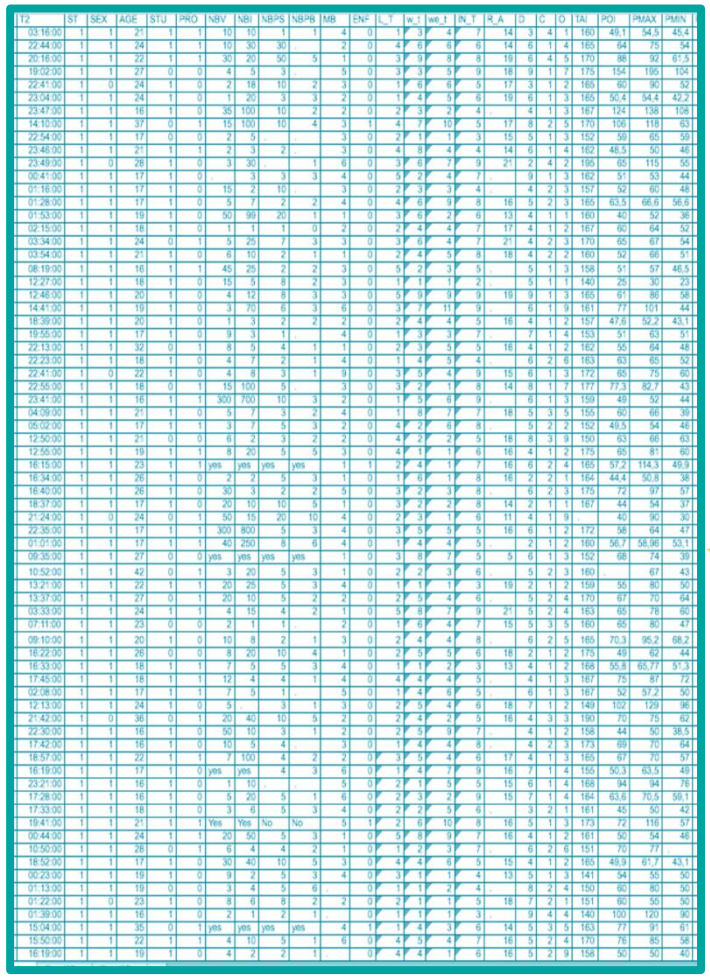
¿Me pagará la deuda?

¿Colombia ganará el Mundial?

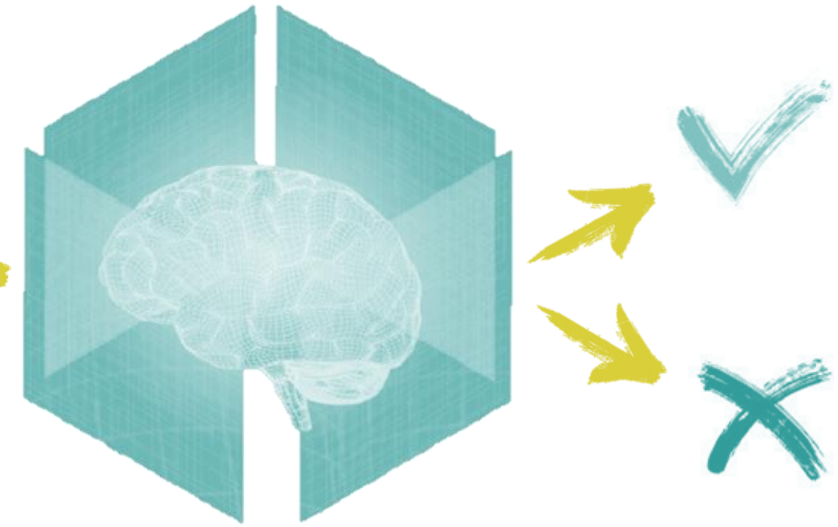
¿Qué variables o características tengo?

Proceso





TZ	ST	SEX	AGE	STU	PRO	NBV	NBI	NBPS	NBPB	MB	ENF	L_T	w_t	we_t	IN_T	R_A	D	IC	D	TAI	POI	PMAX	PMIN
03:16:00	1	1	21	1	1	10	10	1	1	4	0	1	3	4	7	14	3	4	1	160	48,1	54,5	45,4
22:44:00	1	1	24	1	1	10	30	30	.	2	0	4	6	6	6	14	6	1	4	165	64	75	54
20:16:00	1	1	22	1	1	30	20	30	5	1	0	3	9	8	8	19	6	4	5	170	98	92	61,5
19:02:00	1	1	27	0	0	4	5	3	.	5	0	3	3	5	9	18	9	1	7	175	154	195	104
22:41:00	1	0	24	1	0	2	18	10	2	3	0	1	6	6	5	17	3	1	2	165	60	90	52
23:04:00	1	1	24	1	0	1	20	3	3	2	0	1	4	5	6	19	6	1	3	165	50,4	54,4	42,2
23:47:00	1	1	16	1	0	35	100	10	2	2	0	2	3	2	4	.	4	1	3	167	124	138	108
14:10:00	1	1	37	0	1	15	100	10	4	3	1	4	7	10	5	17	8	2	5	170	106	118	63
22:54:00	1	1	17	0	0	2	5	.	.	3	0	2	1	1	3	15	5	1	3	152	59	65	59
23:46:00	1	1	21	1	1	2	3	2	.	3	0	3	4	8	4	14	6	1	4	162	48,5	50	46
23:49:00	1	0	28	1	0	3	30	.	1	6	0	3	6	7	9	21	2	4	2	195	65	115	55
00:41:00	1	1	17	1	0	.	3	3	3	4	0	5	2	4	7	.	9	1	3	162	51	53	44
01:16:00	1	1	17	1	0	15	2	10	.	3	0	2	3	3	4	.	4	2	3	157	52	60	48
01:28:00	1	1	17	1	0	5	7	2	2	4	0	4	6	9	8	16	5	2	3	165	63,5	66,6	56,6
01:53:00	1	1	19	1	0	50	99	20	1	1	0	3	6	2	6	13	4	1	1	160	40	52	38
02:15:00	1	1	18	1	0	1	1	1	0	2	0	2	4	4	7	17	4	1	2	167	60	64	52
03:34:00	1	1	24	0	1	5	25	7	3	3	0	3	6	4	7	21	4	2	3	170	65	67	54
03:54:00	1	1	21	1	0	6	10	2	1	1	0	2	4	5	8	18	4	2	2	160	52	66	51
08:19:00	1	1	16	1	1	45	25	2	2	3	0	5	2	3	5	.	5	1	3	158	51	57	46,5
12:27:00	1	1	18	1	0	15	5	8	2	3	0	1	1	1	2	.	5	1	1	140	25	30	23
12:46:00	1	1	20	1	0	4	12	8	3	3	0	5	9	9	9	19	9	1	3	165	61	86	58
14:41:00	1	1	19	1	0	3	70	6	3	6	0	3	7	11	9	.	6	1	9	161	77	101	44
18:39:00	1	1	20	1	0	1	3	2	2	2	0	2	4	4	5	16	4	1	2	157	47,6	52,2	43,1
19:55:00	1	1	17	1	0	9	3	1	.	4	0	1	3	3	7	.	7	1	4	153	51	63	51
22:13:00	1	1	32	0	1	8	5	4	1	1	0	2	3	5	5	18	4	1	2	162	55	64	48
22:23:00	1	1	18	1	0	4	7	2	1	4	0	1	4	5	4	.	6	2	6	163	63	65	52
22:41:00	1	0	22	1	0	4	8	3	1	9	0	3	5	4	9	15	6	1	3	172	65	75	60
22:55:00	1	1	18	0	1	15	100	5	.	3	0	3	2	1	8	14	8	1	7	177	77,3	82,7	43
23:41:00	1	1	16	1	1	300	700	10	3	2	0	1	5	6	9	.	6	1	3	159	49	52	44
04:09:00	1	1	21	1	0	5	7	3	2	4	0	1	8	7	7	18	5	3	5	155	60	66	39
05:02:00	1	1	17	1	1	3	7	5	3	2	0	4	2	6	8	.	5	2	2	152	49,5	54	46
12:50:00	1	1	21	0	0	8	2	3	2	2	0	4	2	2	5	18	8	3	9	150	63	66	63
12:55:00	1	1	19	1	1	8	20	5	5	3	0	4	1	1	6	16	4	1	2	175	65	81	60
16:15:00	1	1	23	1	1	yes	yes	yes	yes	1	0	2	4	1	1	16	6	2	4	165	57,2	114,3	49,9
16:34:00	1	1	26	1	0	2	2	5	3	1	0	1	6	1	8	18	2	2	1	164	44,4	50,8	38
16:40:00	1	1	26	1	0	30	3	2	2	5	0	3	2	3	8	.	6	2	3	175	72	97	57
18:37:00	1	1	17	1	0	20	10	10	5	1	0	3	2	2	8	14	2	1	1	167	44	54	37
21:24:00	1	0	24	0	1	50	15	20	10	4	0	2	3	1	6	11	4	1	9	.	40	90	30
22:35:00	1	1	17	1	1	300	800	5	3	4	0	3	5	5	5	16	6	1	2	172	58	64	47
01:01:00	1	1	17	1	1	40	250	8	6	4	0	1	4	4	5	.	2	1	2	160	56,7	58,96	53,1
09:35:00	1	1	27	0	0	yes	yes	yes	yes	1	0	3	8	7	5	5	6	1	3	152	68	74	39
10:52:00	1	1	42	0	1	3	20	5	3	1	0	2	2	3	6	.	5	2	3	160	.	67	43
13:21:00	1	1	22	1	1	20	25	5	3	4	0	1	1	1	3	19	2	1	2	159	55	80	50
13:37:00	1	1	27	0	1	20	10	5	2	2	0	2	5	4	6	.	5	2	4	170	67	70	64
03:33:00	1	1	24	1	1	4	15	4	2	1	0	5	8	7	9	21	5	2	4	163	65	78	60
07:11:00	1	1	23	0	0	2	1	1	.	2	0	1	6	4	7	15	5	3	5	160	65	80	47
09:10:00	1	1	20	1	0	10	8	2	1	3	0	2	4	4	8	.	6	2	5	165	70,3	95,2	68,2
16:22:00	1	1	26	0	0	8	20	10	4	1	0	2	5	5	6	18	2	1	2	175	49	62	44
16:33:00	1	1	18	1	1	7	5	5	3	4	0	1	1	2	3	13	4	1	2	168	55,8	65,77	51,3
17:45:00	1	1	18	1	1	12	4	4	1	4	0	4	4	4	5	.	4	1	3	167	75	87	72
02:08:00	1	1	17	1	1	7	5	1	.	5	0	1	4	6	5	.	6	1	3	167	52	57,2	50
12:13:00	1	1	24	1	0	5	.	3	1	3	0	2	5	4	6	18	7	1	2	149	102	129	98
21:42:00	1	0	36	0	1	20	40	10	5	2	0	2	4	2	5	16	4	3	3	190	70	75	62
22:30:00	1	1	16	1	0	50	10	3	1	2	0	2	5	3	7	.	4	1	3	158	44	50	38,5
17:42:00	1	1	16	1	0	10	5	4	.	3	0	1	4	4	8	.	4	2	3	173	69	70	64
18:57:00	1	1	22	1	1	7	100	4	2	2	0	3	5	4	6	17	4	1	3	165	67	70	57
16:19:00	1	1	17	1	0	yes	yes	4	3	6	0	1	4	7	9	16	7	1	4	155	50,3	63,5	49
23:21:00	1	1	16	1	0	1	10	.	.	5	0	2	1	5	5	15	6	1	4	168	94	94	76
17:28:00	1	1	16	1	0	5	20	5	1	6	0	2	3	2	9	15	7	1	4	164	63,6	70,5	59,1
17:33:00	1	1	18	1	0	3	6	5	3	4	0	2	2	5	6	.	3	2	1	161	45	50	42
19:41:00	1	1	21	1	1	Yes	Yes	No	No	5	1	2	6	10	8	16	5	1	3	173	72	116	57
00:44:00	1	1	24	1	1	20	50	5	3	1	0	5	8	9	7	16	4	1	2	161	50	54	46
10:50:00	1	1	28	0	1	6	4	4	2	1	0	1	2	3	7	.	6	2	6	151	70	77	.
18:52:00	1	1	17	1	0	30	40	10	5	3	0	4	4	6	5	15	4	1	2	165	49,9	61,7	43,1
00:23:00	1	1	19	1	0	9	2	5	3	4	0	3	1	1	4	13	5	1	3	141	54	55	50
01:13:00	1	1	19	0	0	3	4	5	6	.	0	2	1	2	4	.	8	2	4	150	60	80	50
01:22:00	1	0	23	1	0	8	6	8	2	2	0	2	1	1	5	18	7	2	1	151	60	55	50
01:39:00	1	1	16	1	0	2	1	2	1	.	0	2	1	1	3	.	9	4	4	140	100	120	90
15:04:00	1	1	35	0	1	yes	yes	yes	yes	4	1	1	4	3	6	14	5	3	5	163	77	91	61
15:50:00	1	1	22	1	1	4	10	5	1	6	0	4	5	4	7	18	5	2	4	170	76	85	56
16:19:00	1	1	19	1	0	4	2	2	1	.	0	4	4	1	6	16	5	2	9	158	50	50	40



TSFresh

+ de 800 Características de la serie de Tiempo

Parte 1

maximum(x)	highest value of the time series x.
mean(x)	mean of x
mean_abs_change(x)	mean over the absolute differences between subsequent time series values which is
mean_change(x)	mean over the absolute differences between subsequent time series values which is
median(x)	median of x
minimum(x)	lowest value of the time series x.
standard_deviation(x)	standard deviation of x
variance(x)	variance of x
kurtosis(x)	kurtosis of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G2).
skewness(x)	sample skewness of x (calculated with the adjusted Fisher-Pearson standardized moment coefficient G1).
sum_values(x)	sum over the time series values
length(x)	length of x
first_location_of_maximum(x)	first location of the maximum value of x.
first_location_of_minimum(x)	first location of the minimal value of x.
last_location_of_maximum(x)	relative last location of the maximum value of x.
last_location_of_minimum(x)	last location of the minimal value of x.
count_above_mean(x)	number of values in x that are higher than the mean of x
count_below_mean(x)	number of values in x that are lower than the mean of x
longest_strike_above_mean(x)	length of the longest consecutive subsequence in x that is bigger than the mean of x
longest_strike_below_mean(x)	length of the longest consecutive subsequence in x that is smaller than the mean of x

TSFresh

+ de 800 Características de la serie de Tiempo

Parte 2

linear_trend(x, param)	linear least-squares regression for the values of the time series
agg_linear_trend(x, param)	linear least-squares regression for values of the time series that were aggregated over chunks.
quantile(x, q)	q quantile of x.
number_peaks(x, n)	number of peaks of at least support n in the time series x.
abs_energy(x)	absolute energy of the time series which is the sum over the squared values
energy_ratio_by_chunks(x, param)	sum of squares of chunk (10_segments)
absolute_sum_of_changes(x)	sum over the absolute value of consecutive changes in the series x
change_quantiles(x, ql, qh, isabs, f_agg)	First fixes a corridor given by the quantiles ql and qh of the distribution of x.
percentage_of_reoccurring_datapoints_to_all_data points(x)	percentage of unique values, that are present in the time series more than once.
percentage_of_reoccurring_values_to_all_values(x)	ratio of unique values, that are present in the time series more than once.
range_count(x, min, max)	Count observed values within the interval [min, max).
value_count(x, value)	Count occurrences of value in time series x.
sum_of_reoccurring_data_points(x)	sum of all data points, that are present in the time series more than once.
sum_of_reoccurring_values(x)	sum of all values, that are present in the time series more than once.
has_duplicate(x)	Checks if any value in x occurs more than once
has_duplicate_max(x)	Checks if the maximum value of x is observed more than once
has_duplicate_min(x)	Checks if the minimal value of x is observed more than once
approximate_entropy(x, m, r)	Implements a vectorized Approximate entropy algorithm.
sample_entropy(x)	Calculate and return sample entropy of x.
binned_entropy(x, max_bins)	First bins the values of x into max_bins equidistant bins.

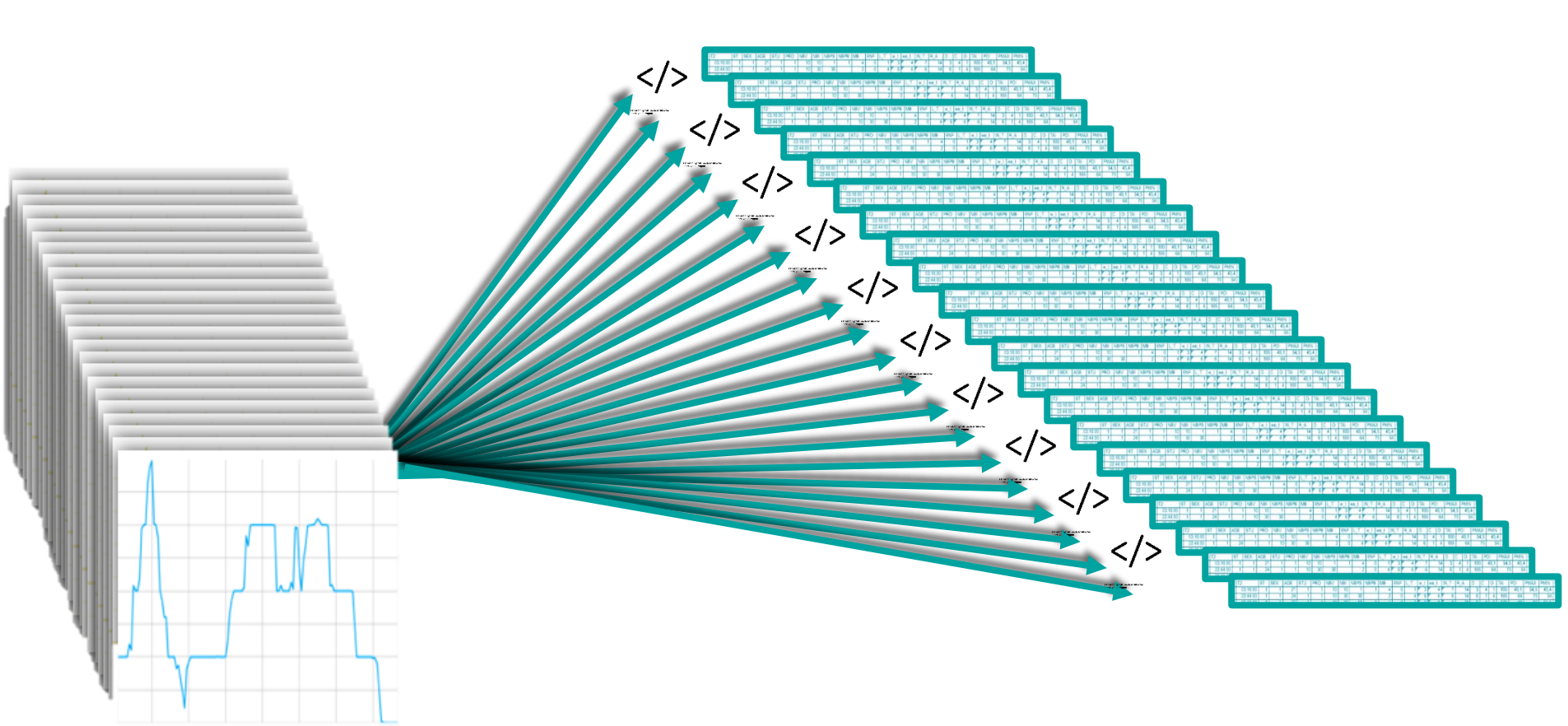
TSFresh

+ de 800 Características de la serie de Tiempo

Parte 3

ar_coefficient(x, param)	This feature calculator fits the unconditional maximum likelihood of an autoregressive AR(k) process.
agg_autocorrelation(x, param)	value of an aggregation function f_agg (e.g.
autocorrelation(x, lag)	autocorrelation of the specified lag, according to the formula [1]
partial_autocorrelation(x, param)	value of the partial autocorrelation function at the given lag.
augmented_dickey_fuller(x, param)	The Augmented Dickey-Fuller test is a hypothesis test which checks whether a unit root is present.
c3(x, lag)	Measure of non linearity in the time series.
cwt_coefficients(x, param)	Continuous wavelet transform for the Ricker wavelet, also known as the “Mexican hat wavelet” which is
fft_coefficient(x, param)	fourier coefficients of the one-dimensional discrete Fourier Transform for real input by fast
friedrich_coefficients(x, param)	Coefficients of polynomial , which has been fitted to
index_mass_quantile(x, param)	Those apply features calculate the relative index i where q% of the mass of the time series x lie left of i.
large_standard_deviation(x, r)	Boolean variable denoting if the standard dev of x is higher than ‘r’*(max - min of x.) "r=0.05:1"
variance_larger_than_standard_deviation(x)	Boolean variable denoting if the variance of x is greater than its standard deviation.
symmetry_looking(x, param)	Boolean variable denoting if the distribution of x looks symmetric.
max_langevin_fixed_point(x, r, m)	Largest fixed point of dynamics :math:argmax_x {h(x)=0}` estimated from polynomial ,
number_crossing_m(x, m)	number of crossings of x on m.
number_cwt_peaks(x, n)	This feature calculator searches for different peaks in x.
ratio_beyond_r_sigma(x, r)	Ratio of values that are more than r*std(x) (so r sigma) away from the mean of x.
ratio_value_number_to_time_series_length(x)	factor which is 1 if all values in the time series occur only once, and below one if this is not the case.
set_property(key, value)	This method decorator that sets the property key of the function to value
spkt_welch_density(x, param)	Cross power spectral density of the time series x at different frequencies.
time_reversal_asymmetry_statistic(x, lag)	Proposed by Fulcher and Jones as a promising feature to extract from time series.

Construcción de características de la ts

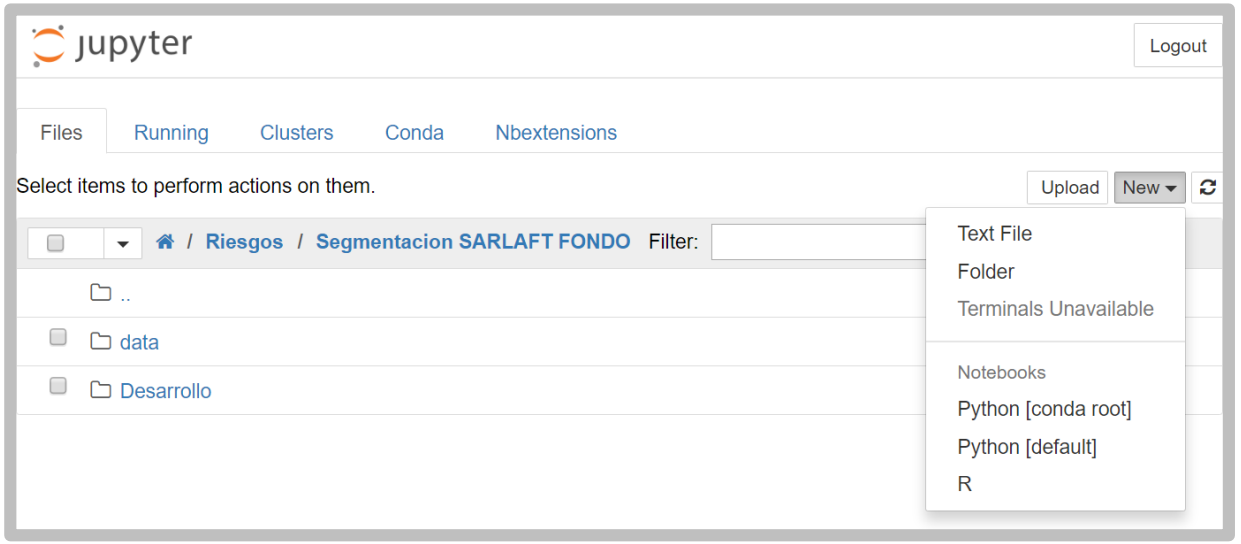


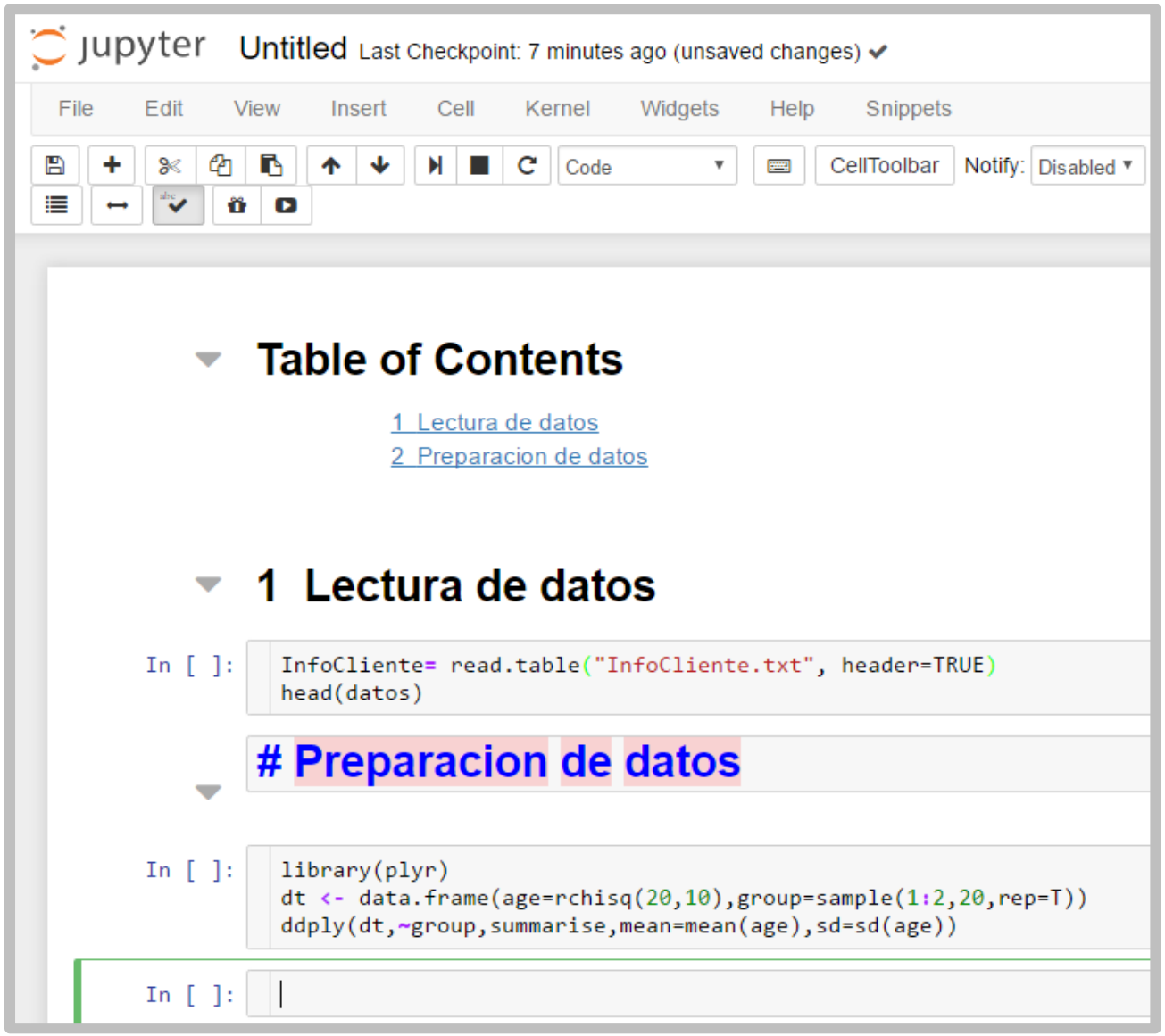
Construcción de características de la ts

Algoritmo resumen mediante código Spark

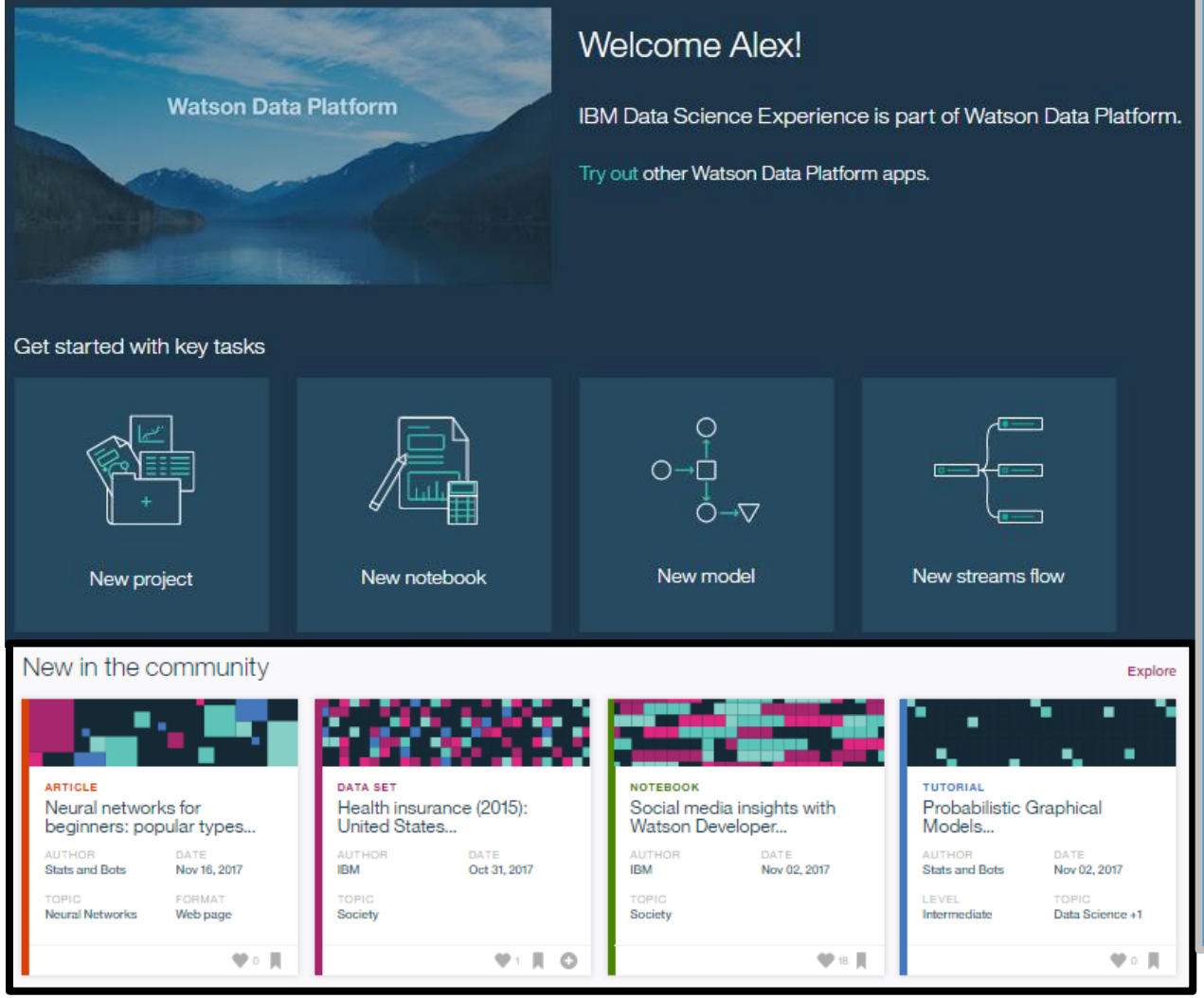
```
DB = spark.read(…)
#[Row(Hash='id1', valor=100', periodo='201208'), Row(…), …   ]
DB = DB.groupBy(Hash).agg(F.collect_list(valor), F.collect_list(periodo))
#[Row(Hash='id1', valor=[100,105,110,…]', periodo=['201208', '201209', '201210',…], Row(…),…]


def Create_Features(pyspark_row):
        ts = create_pandas(pyspark_row)
        ts = fix_time_series(ts)

        f_fuga = tsfresh.extract_features(ts, column_id='id', column_sort='periodo', n_jobs=0)\
                .T.id.to_dict()

        r = {'Hash':pyspark_row.Hash, 'features_fuga':f_fuga}

        return  Row(**r)

caracteristicas = DB.rdd.map(lambda x: all_features_entrenamiento(x))
```

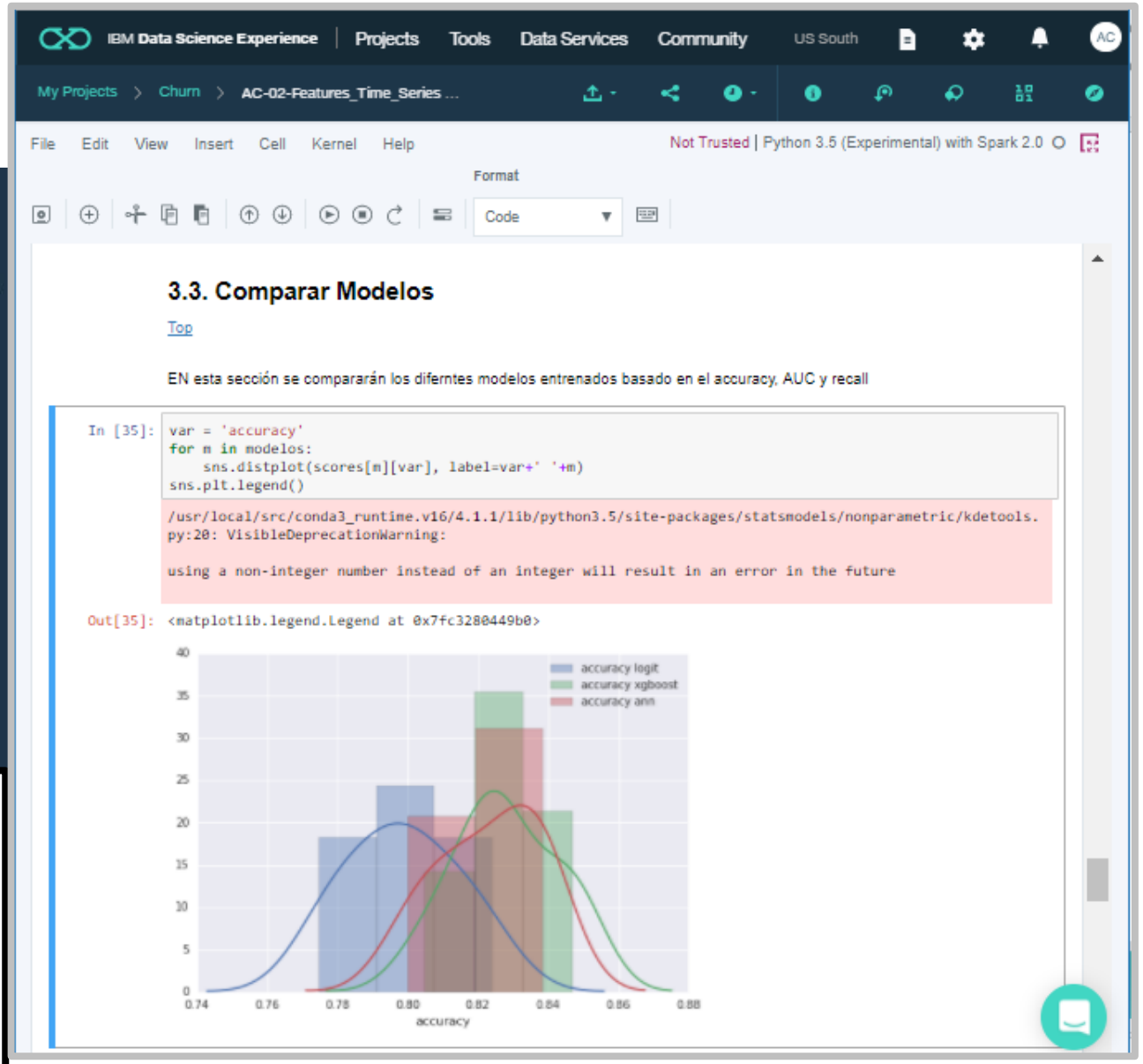
Project Jupyter exists to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.



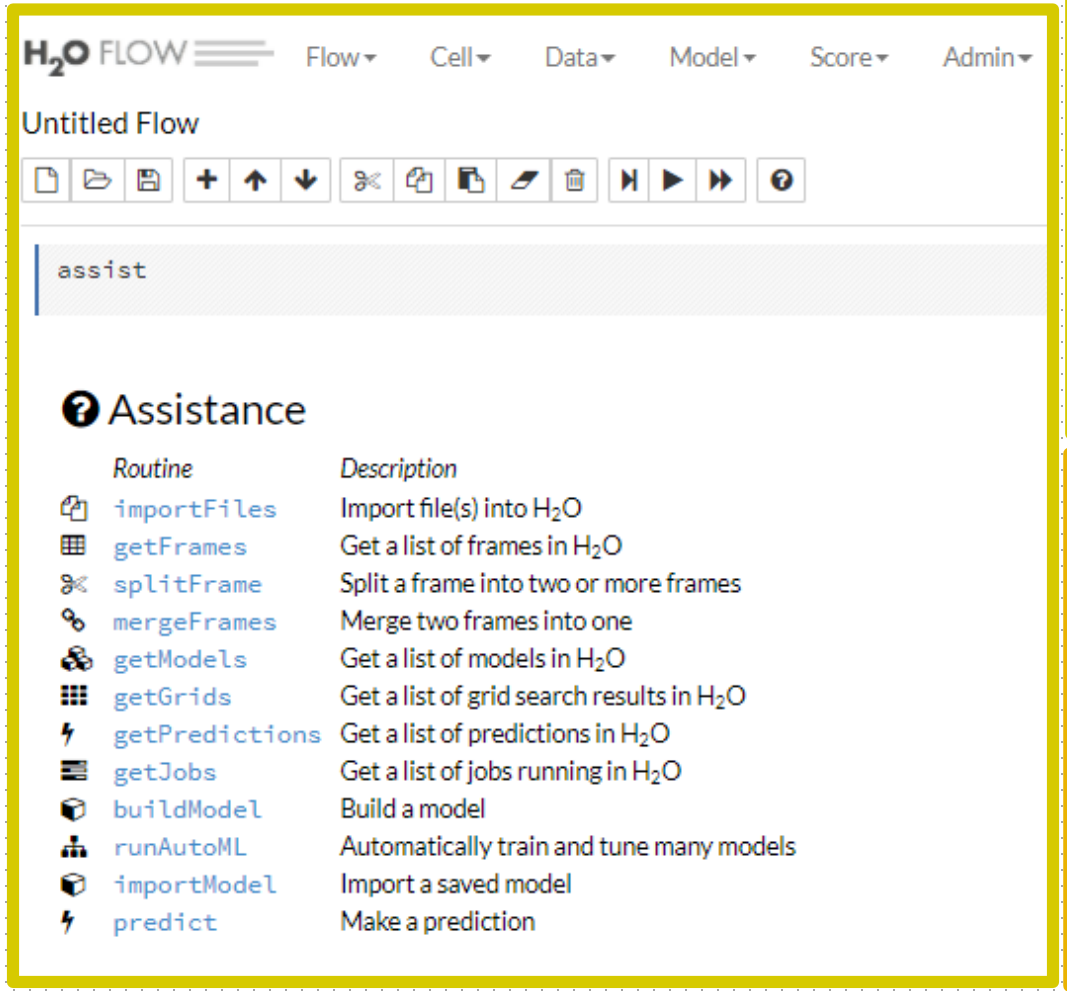


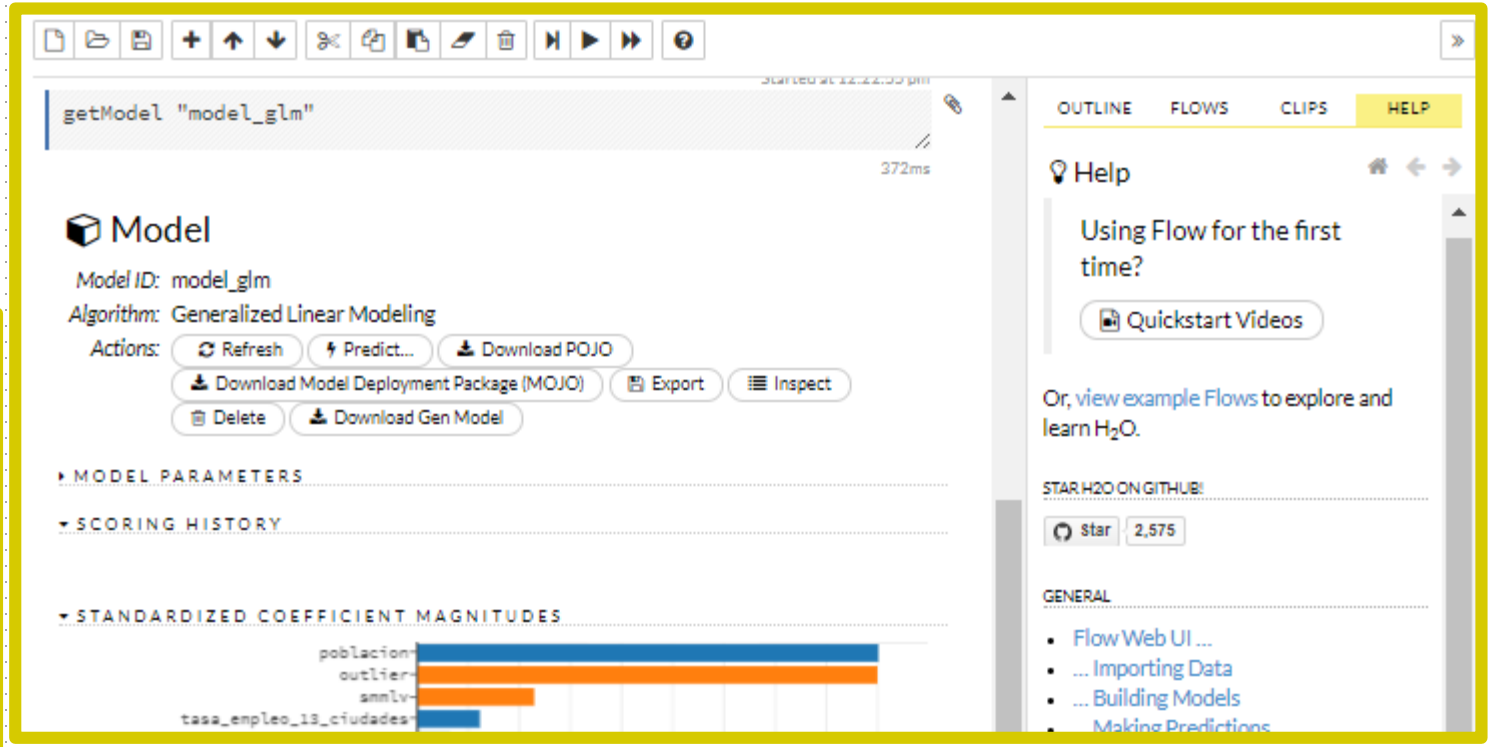
IBM Data Science Experience

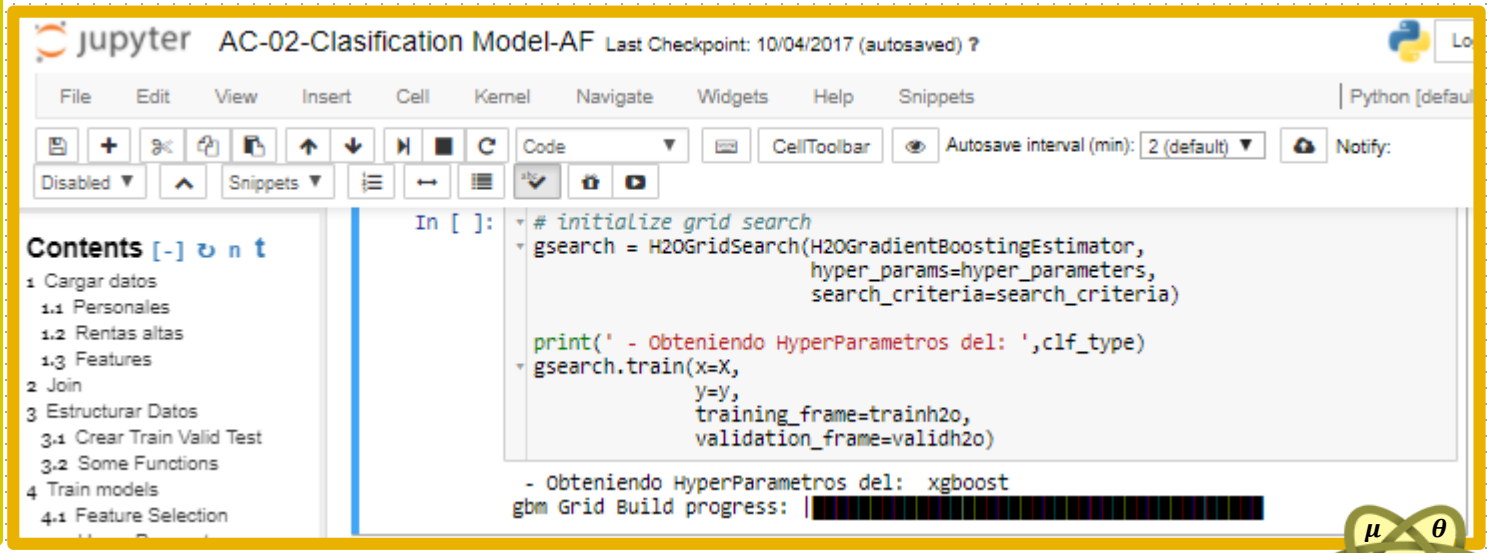




H2O.ai


H2O FLOW
Flow
Cell
Data
Model
Score
Admin
Untitled Flow
assist
Assistance
Routine
Description
importFiles
Import file(s) into H2O
getFrames
Get a list of frames in H2O
splitFrame
Split a frame into two or more frames
mergeFrames
Merge two frames into one
getModels
Get a list of models in H2O
getGrids
Get a list of grid search results in H2O
getPredictions
Get a list of predictions in H2O
getJobs
Get a list of jobs running in H2O
buildModel
Build a model
runAutoML
Automatically train and tune many models
importModel
Import a saved model
predict
Make a prediction


getModel "model_glm"
372ms
Model
Model ID: model_glm
Algorithm: Generalized Linear Modeling
Actions:
Refresh
Predict...
Download POJO
Download Model Deployment Package (MOJO)
Export
Inspect
Delete
Download Gen Model
MODEL PARAMETERS
SCORING HISTORY
STANDARDIZED COEFFICIENT MAGNITUDES
poblacion
outlier
anmlv
tasa_empleo_13_ciudades
OUTLINE
FLOWS
CLIPS
HELP
Help
Using Flow for the first time?
Quickstart Videos
Or, view example Flows to explore and learn H2O.
STAR H2O ON GITHUB!
Star
2,575
GENERAL
Flow Web UI ...
... Importing Data
... Building Models
... Making Predictions


jupyter
AC-02-Clasification Model-AF
Last Checkpoint: 10/04/2017 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Navigate
Widgets
Help
Snippets
Code
CellToolbar
Autosave interval (min):
2 (default)
Notify:
Disabled
Snippets
Contents
1 Cargar datos
1.1 Personales
1.2 Rentas altas
1.3 Features
2 Join
3 Estructurar Datos
3.1 Crear Train Valid Test
3.2 Some Functions
4 Train models
4.1 Feature Selection
In [ ]:
# initialize grid search
gsearch = H2OGridSearch(H2OGradientBoostingEstimator,
hyper_params=hyper_parameters,
search_criteria=search_criteria)
print(' - Obteniendo HyperParametros del: ',clf_type)
gsearch.train(x=X,
y=y,
training_frame=trainh2o,
validation_frame=validh2o)
- Obteniendo HyperParametros del: xgboost
gbm Grid Build progress:

Construcción de características

Algoritmo resumen H2O mediante código python

```
baseh2o = h2o.H2OFrame(base.values.tolist(), column_names=base.columns.values.tolist())

y = 'cliente_fuga'
X = selected_features


#Grid search
hyper_parameters = {'ntrees':[100,500,...],'max_depth':[4,5,...], 'sample_rate':np.arange(0.3,0.6,...),
'col_sample_rate':np.arange(0.3,0.6,...), ...}
search_criteria = {'strategy':'RandomDiscrete', 'max_runtime_secs':10800,'seed': 1234}
gsearch = H2OGridSearch(H2OGradientBoostingEstimator, hyper_params=hyper_parameters, search_criteria=search_criteria)
gsearch.train(x=X, y=y, training_frame=trainh2o, validation_frame=validh2o)


#Train Model
model_h2o = H2OGradientBoostingEstimator(ntrees=700, max_depth=5, col_sample_rate=0.7, col_sample_rate_per_tree=0.9, \
                                         sample_rate=0.9, learn_rate=0.01, min_rows=50, seed=1234)
model_h2o.train(X, y, training_frame=train_completoh2o, validation_frame=testh2o)
model_h2o.model_id='Modelo_Fuga_h2o'
```

Alex Castaño Ballesteros

github.com/acastanob

linkedin.com/in/acastanob/

acastanob@gmail.com

312 840 88 87

GRACIAS