

Prediciendo la contaminación del aire en Madrid

María Medina - PiperLab
@mariamedp



PIPERLAB



Hello my name is



María Medina

Data Scientist en PiperLab

Coorganizadora PyLadies Madrid

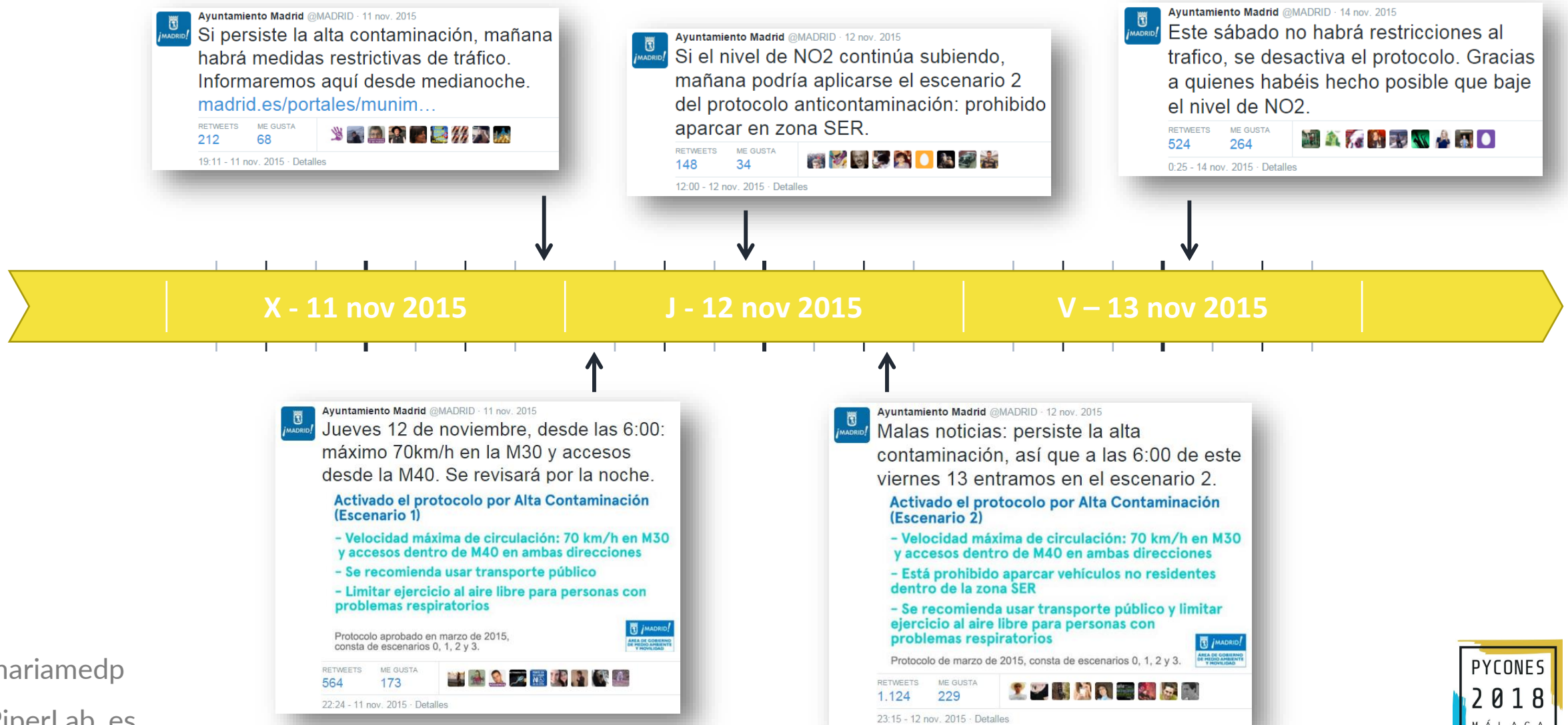


@mariamedp

@PiperLab_es



Érase una vez...





Datos de contaminación

Portal de Datos Abiertos del Ayuntamiento de Madrid



Calidad del aire: Datos en tiempo real

Calidad del aire: Datos en tiempo real. El Sistema Integral de la Calidad del Aire del Ayuntamiento de Madrid permite conocer en cada momento los niveles de contaminación atmosférica en el municipio. En este conjunto de datos puede obtener la información actualizada en tiempo real, actualizándose estos datos cada hora, y...

Calidad del aire: Estaciones de control

Calidad del aire: Estaciones de control. El Sistema de Vigilancia está formado por 24 Estaciones Remotas automáticas que recogen la información básica para la vigilancia atmosférica. Poseen los analizadores necesarios para la medida correcta de los niveles de gases y de partículas. Las estaciones remotas son de varios tipos: Urbanas de fondo; representativas de la exposición...

Calidad del aire: Datos horarios años 2003 a 2016

Calidad del aire: Datos horarios años 2003 a 2016. El Sistema Integral de la Calidad del Aire del Ayuntamiento de Madrid permite conocer en cada momento los niveles de contaminación atmosférica en el municipio. En este conjunto de datos puede obtener la información recogida por las estaciones de control de

Calidad del aire: Datos diarios años 2003 a 2016

Calidad del aire: Datos diarios años 2003 a 2016. El Sistema Integral de la Calidad del Aire del Ayuntamiento de Madrid permite conocer en cada momento los niveles de contaminación atmosférica en el municipio. En este conjunto de datos puede obtener la información recogida por las estaciones de control de

@mariamedp

@PiperLab_es



Cargar datos con pandas



```
col_horas = ["{}_{}".format(h, tipo) for h in range(1, 25) for tipo in ("medida", "valido")]

datos = pd.read_csv("http://www.mambiente.munimadrid.es/opendata/horario.txt",
                    dtype="str",
                    names=["estacion_parte1", "estacion_parte2", "estacion_parte3",
                          "parametro", "tecnic", "periodo", "yyyy", "mm", "dd"] + col_horas)

datos.head()
```

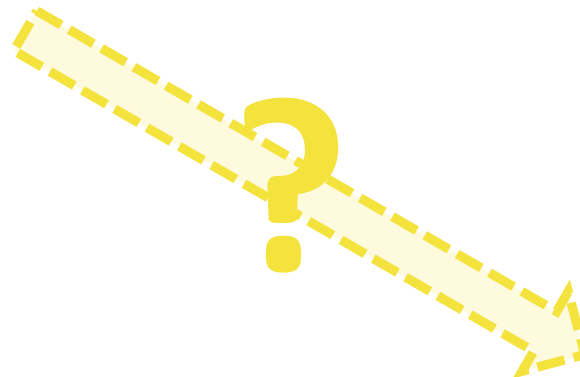
	estacion_parte1	estacion_parte2	estacion_parte3	parametro	tecnic	periodo	yyyy	mm	dd	1_medida	...	20_medida	20_valido	21_medida	21_valido
0	28	079	004	01	38	02	2018	09	17	00009	...	00000	N	00000	N
1	28	079	004	06	48	02	2018	09	17	000.5	...	00000	N	00000	N
2	28	079	004	07	08	02	2018	09	17	00021	...	00000	N	00000	N
3	28	079	004	08	08	02	2018	09	17	00100	...	00000	N	00000	N
4	28	079	004	12	08	02	2018	09	17	00133	...	00000	N	00000	N



Procesar datos con pandas



<estación, fecha, contaminante>	1_medida	1_valido	...	24_medida	24_valido
...	xxx	V/N	...	yyy	V/N



<estación, fecha, contaminante>	hora	medida	valido
...	1	xxx	V/N
...
...	24	yyy	V/N

Procesar datos con pandas



<estación, fecha, contaminante>	1_medida	1_valido	...	24_medida	24_valido
...	xxx	V/N	...	yyy	V/N

<estación, fecha, contaminante>	hora	medida	valido
...	1	xxx	V/N
...
...	24	yyy	V/N



Procesar datos con pandas



<estación, fecha, contaminante>	1_medida	1_valido	...	24_medida	24_valido
...	xxx	V/N	...	yyy	V/N

df.melt()

<estación, fecha, contaminante>	variable	valor
	1_medida	xxx
	1_valido	V/N

...	24_medida	yyy
	24_valido	V/N

<estación, fecha, contaminante>	hora	medida	valido
...	1	xxx	V/N
...
...	24	yyy	V/N





Procesar datos con pandas



<estación, fecha, contaminante>	1_medida	1_valido	...	24_medida	24_valido
...	xxx	V/N	...	yyy	V/N

df.melt()

<estación, fecha, contaminante>	variable	valor
	1_medida	xxx
	1_valido	V/N

...	24_medida	yyy
	24_valido	V/N

str.split("_")

<estación, fecha, contaminante>	hora	variable	valor
	1	medida	xxx
	1	valido	V/N

...	24	medida	yyy
	24	valido	V/N

<estación, fecha, contaminante>	hora	medida	valido
...	1	xxx	V/N
...
...	24	yyy	V/N





Procesar datos con pandas



<estación, fecha, contaminante>	1_medida	1_valido	...	24_medida	24_valido
...	xxx	V/N	...	yyy	V/N

df.melt()

<estación, fecha, contaminante>	variable	valor
	1_medida	xxx
	1_valido	V/N

...	24_medida	yyy
	24_valido	V/N

str.split("_")

<estación, fecha, contaminante>	hora	variable	valor
	1	medida	xxx
	1	valido	V/N

...	24	medida	yyy
	24	valido	V/N

pd.pivot_table(df)

<estación, fecha, contaminante>	hora	medida	valido
...	1	xxx	V/N
...
...	24	yyy	V/N





Datóxido de Nitrógeno

 @datoxnitro_bot



Datóxido Nitrógeno
@datoxnitro_bot

Bot que publica en tiempo real los niveles de contaminación por dióxido de nitrógeno en la ciudad de Madrid. Datos de datos.madrid.es.

📍 Madrid, Comunidad de Madrid

Tweets 20,2 mil Siguiendo 2 Seguidores 1.676 Siguiendo

Tweets Tweets y respuestas Multimedia

📌 Tweet fijado

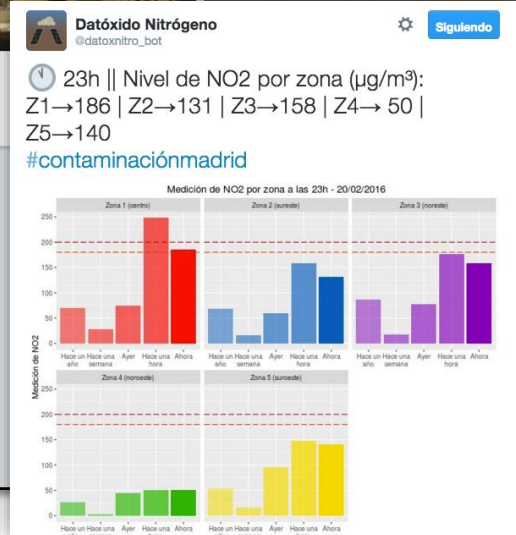
Datóxido Nitrógeno @datoxnitro_bot · 25 ene. 2016
¡Lanzamos el bot coincidiendo con la charla en #databeers #mad!
Hilo explicando el protocolo #contaminaciónmadrid

Datóxido Nitrógeno @datoxnitro_bot
En Madrid hay 24 estaciones que miden la concentración de dióxido de nitrógeno (NO2) en el aire. Mapa ubicaciones: [mapa](#)



Datóxido de Nitrógeno

 @datoxnitro_bot



Tweets

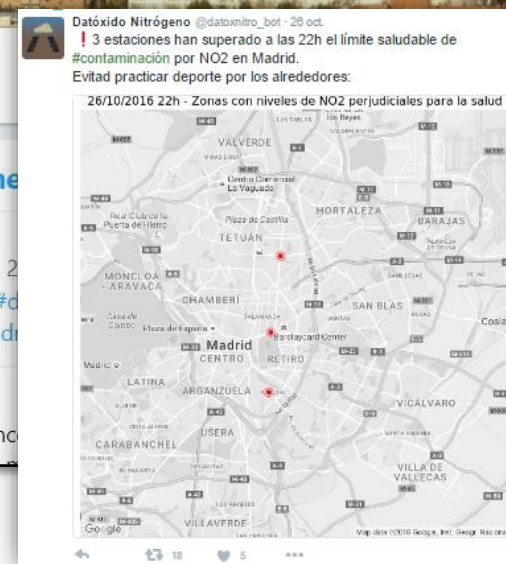
Tweets y respuestas

Multimedia

Tweet fijado

Datóxido Nitrógeno @datoxnitro_bot · 25 ene. 2016
¡Lanzamos el bot coincidiendo con la charla en #PyConES18! Hilo explicando el protocolo #contaminaciónmadrid

Datóxido Nitrógeno @datoxnitro_bot
En Madrid hay 24 estaciones que miden la concentración de dióxido de nitrógeno (NO₂) en el aire. Mapa de ubicación de las estaciones: [https://bit.ly/2wvzqzq](#)



Datóxido Nitrógeno
@datoxnitro_bot

ÚLT. HORA ⚠️ Alcanzado el Escenario 1 por alta #contaminaciónmadrid.
El Ayto. de @MADRID podría activar restricciones para el viernes 05ago

@mariamedp

@PiperLab_es

Tuitear con Twython



Twython 

```
from twython import Twython  
  
api = Twython(CONSUMER_KEY, CONSUMER_SECRET, ACCESS_KEY, ACCESS_SECRET)
```

```
api.update_status(status = text)
```

```
api.update_status(status = text,  
                  in_reply_to_status_id = tweet_id)
```

```
with open(mediafile, 'rb') as f:  
    ret = api.upload_media(media=f)  
  
api.update_status(status = text,  
                  media_ids = ret["media_id"])
```

@mariamedp

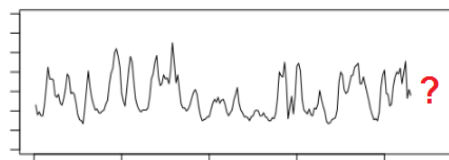
@PiperLab_es



¿Podemos predecir la contaminación?



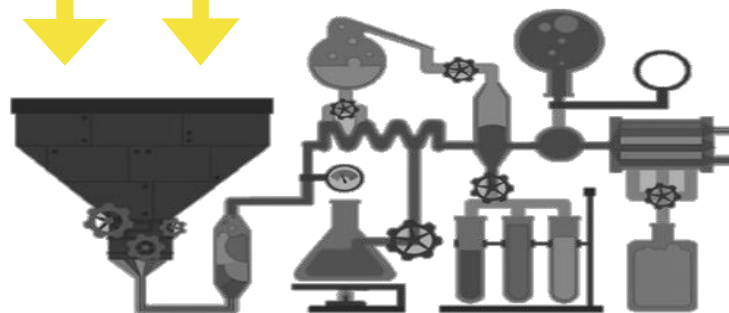
Proceso de predicción



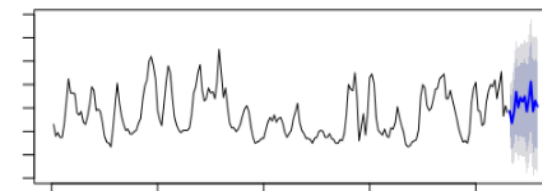
Histórico
contaminación

Variables externas

Meteorología,
Calendarios,
Tráfico, ...



Entrenamiento: algoritmo que
detecta **patrones** en los datos

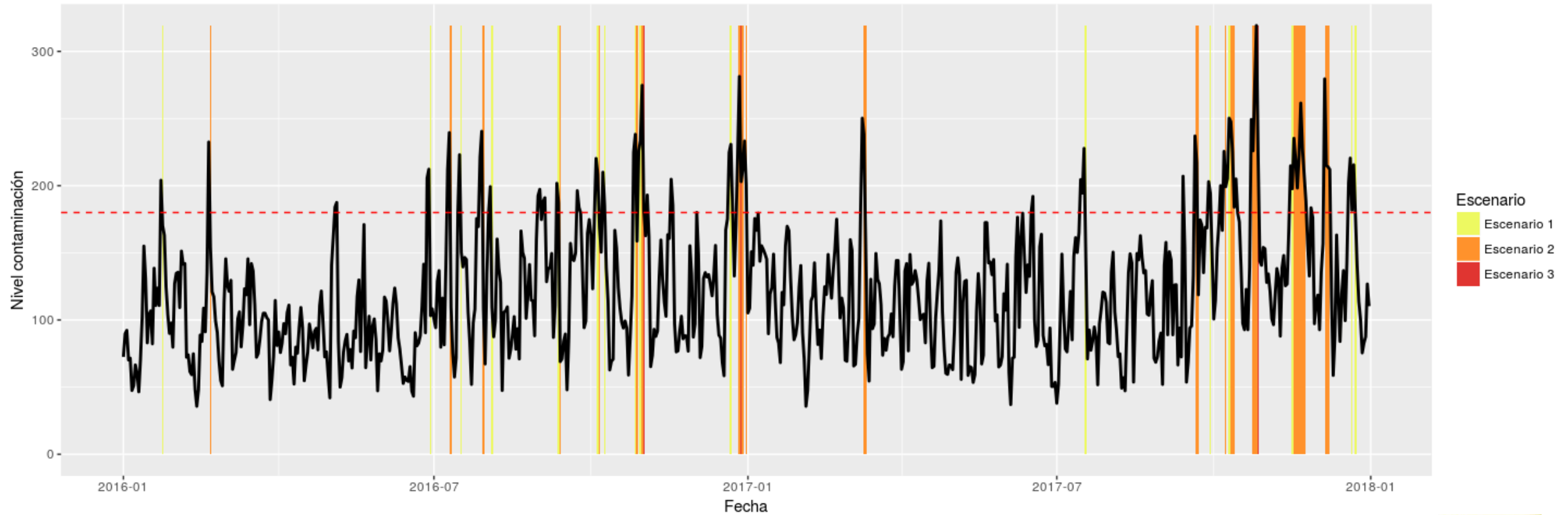


Predicción



Serie temporal “resumen”

Objetivo: saber cuándo se va a activar el protocolo



@mariamedp

@PiperLab_es



Modelos ARIMA base



ARIMA = Autoregressive Integrated Moving Average



Modelos ARIMA base



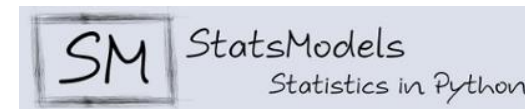
ARIMA = Autoregressive Integrated Moving Average



Aprende de
observaciones anteriores



Modelos ARIMA base



ARIMA = Autoregressive Integrated Moving Average



Aprende de
observaciones anteriores



Aprende de las diferencias
con observaciones anteriores



Modelos ARIMA base



ARIMA = Autoregressive Integrated Moving Average

↖
Aprende de
observaciones anteriores

↑
Aprende de las diferencias
con observaciones anteriores

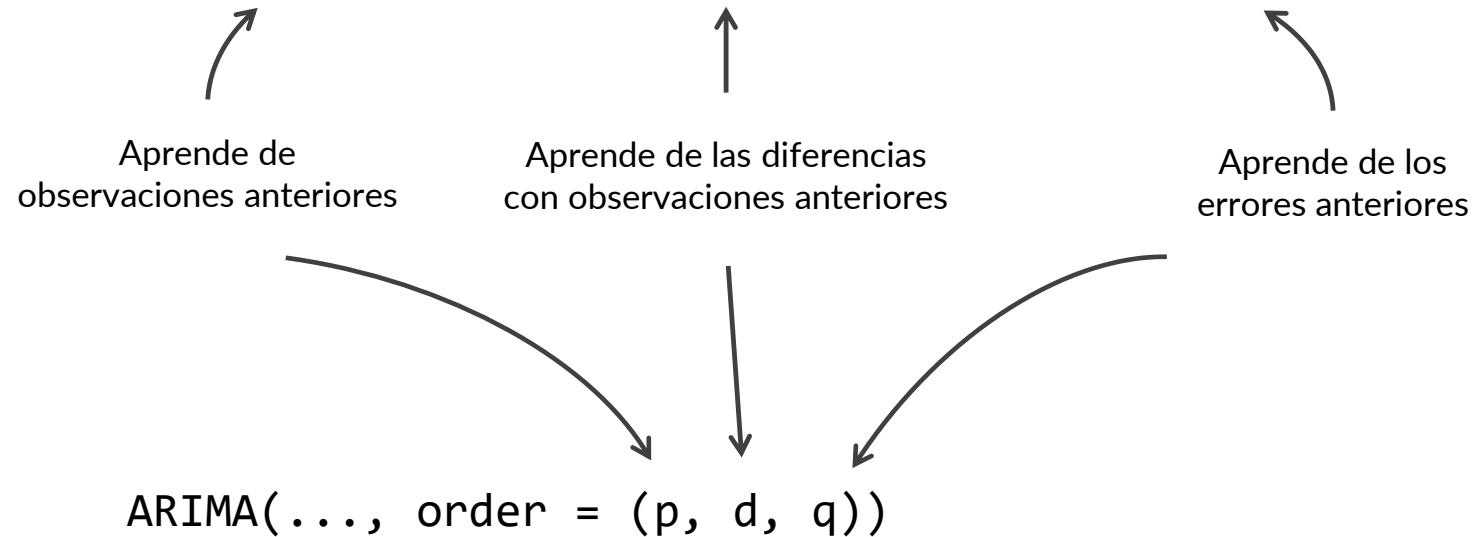
↖
Aprende de los
errores anteriores



Modelos ARIMA base



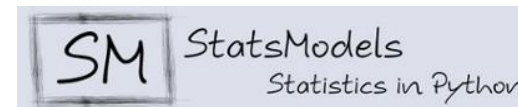
ARIMA = Autoregressive Integrated Moving Average





Modelos ARIMA base

Entrenar modelo



	medida
fecha	
2014-01-01	57.75
2014-01-02	75.75
2014-01-03	82.75
2014-01-04	57.25
2014-01-05	67.00
2014-01-06	87.50
2014-01-07	107.25
2014-01-08	135.75
2014-01-09	188.00
2014-01-10	188.75

```
from statsmodels.tsa.arima_model import ARIMA

model_conf = ARIMA(seriecont, order=(2,1,3))
model_fit = model_conf.fit()
model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.medida	No. Observations:	1460
Model:	ARIMA(2, 1, 3)	Log Likelihood	-7237.366
Method:	css-mle	S.D. of innovations	34.386
Date:	Tue, 18 Sep 2018	AIC	14488.732
Time:	16:02:10	BIC	14525.735
Sample:	01-02-2014	HQIC	14502.536
	- 12-31-2017		

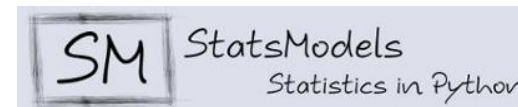
@mariamedp

@PiperLab_es



Modelos ARIMA base

Entrenar modelo



	medida
fecha	
2014-01-01	57.75
2014-01-02	75.75
2014-01-03	82.75
2014-01-04	57.25
2014-01-05	67.00
2014-01-06	87.50
2014-01-07	107.25
2014-01-08	135.75
2014-01-09	188.00
2014-01-10	188.75

```
from statsmodels.tsa.arima_model import ARIMA

model_conf = ARIMA(seriecont, order=(2,1,3))
model_fit = model_conf.fit()
model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.medida	No. Observations:	1460
Model:	ARIMA(2, 1, 3)	Log Likelihood	-7237.366
Method:	css-mle	S.D. of innovations	34.386
Date:	Tue, 18 Sep 2018	AIC	14488.732
Time:	16:02:10	BIC	14525.735
Sample:	01-02-2014	HQIC	14502.536
	- 12-31-2017		

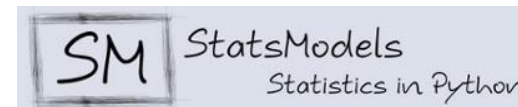
@mariamedp

@PiperLab_es



Modelos ARIMA base

Entrenar modelo



	medida
fecha	
2014-01-01	57.75
2014-01-02	75.75
2014-01-03	82.75
2014-01-04	57.25
2014-01-05	67.00
2014-01-06	87.50
2014-01-07	107.25
2014-01-08	135.75
2014-01-09	188.00
2014-01-10	188.75

```
from statsmodels.tsa.arima_model import ARIMA

model_conf = ARIMA(seriecont, order=(2,1,3))
model_fit = model_conf.fit()
model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.medida	No. Observations:	1460
Model:	ARIMA(2, 1, 3)	Log Likelihood	-7237.366
Method:	css-mle	S.D. of innovations	34.386
Date:	Tue, 18 Sep 2018	AIC	14488.732
Time:	16:02:10	BIC	14525.735
Sample:	01-02-2014	HQIC	14502.536
	- 12-31-2017		

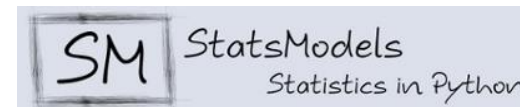
@mariamedp

@PiperLab_es



Modelos ARIMA base

Entrenar modelo



	medida
fecha	
2014-01-01	57.75
2014-01-02	75.75
2014-01-03	82.75
2014-01-04	57.25
2014-01-05	67.00
2014-01-06	87.50
2014-01-07	107.25
2014-01-08	135.75
2014-01-09	188.00
2014-01-10	100.75

```
from statsmodels.tsa.arima_model import ARIMA

model_conf = ARIMA(seriecont, order=(2,1,3))
model_fit = model_conf.fit()
model_fit.summary()
```

ARIMA Model Results

Dep. Variable:	D.medida	No. Observations:	1460
Model:	ARIMA(2, 1, 3)	Log Likelihood	-7237.366
Method:	css-mle	S.D. of innovations	34.386
Date:	Tue, 18 Sep 2018	AIC	14488.732
Time:	16:02:10	BIC	14525.735
Sample:	01-02-2014	HQIC	14502.536
	- 12-31-2017		

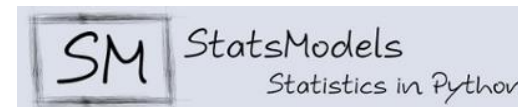
@mariamedp

@PiperLab_es



Modelos ARIMA base

Predecir todo seguido



```
ini_forecast = pd.Timestamp("2017-01-01", freq="D")
ndays = 7

end_forecast = ini_forecast + ndays - 1
```

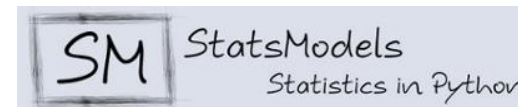
```
model_conf = ARIMA(seriecont[:ini_forecast-1], order=(2,1,2), freq="D")
model_fit = model_conf.fit()

forecast = model_fit.forecast(ndays)[0]
forecast = pd.DataFrame(forecast, columns=["forecast"],
                        index=seriecont[ini_forecast:end_forecast].index)
```




Modelos ARIMA base

Predecir todo seguido



```
ini_forecast = pd.Timestamp("2017-01-01", freq="D")
ndays = 7

end_forecast = ini_forecast + ndays - 1
```

```
model_conf = ARIMA(seriecont[:ini_forecast-1], order=(2,1,2), freq="D")
model_fit = model_conf.fit()

forecast = model_fit.forecast(ndays)[0]
forecast = pd.DataFrame(forecast, columns=["forecast"],
                        index=seriecont[ini_forecast:end_forecast].index)
```



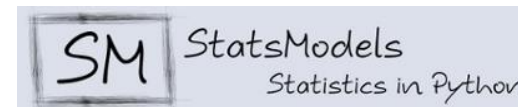
Modelos ARIMA base

Predecir todo seguido



```
ini_forecast = pd.Timestamp("2017-01-01", freq="D")
ndays = 7

end_forecast = ini_forecast + ndays - 1
```



```
model_conf = ARIMA(seriecont[:ini_forecast-1], order=(2,1,2), freq="D")
model_fit = model_conf.fit()

forecast = model_fit.forecast(ndays)[0]
forecast = pd.DataFrame(forecast, columns=["forecast"],
                        index=seriecont[ini_forecast:end_forecast].index)
```



Modelos ARIMA base

Predecir todo seguido

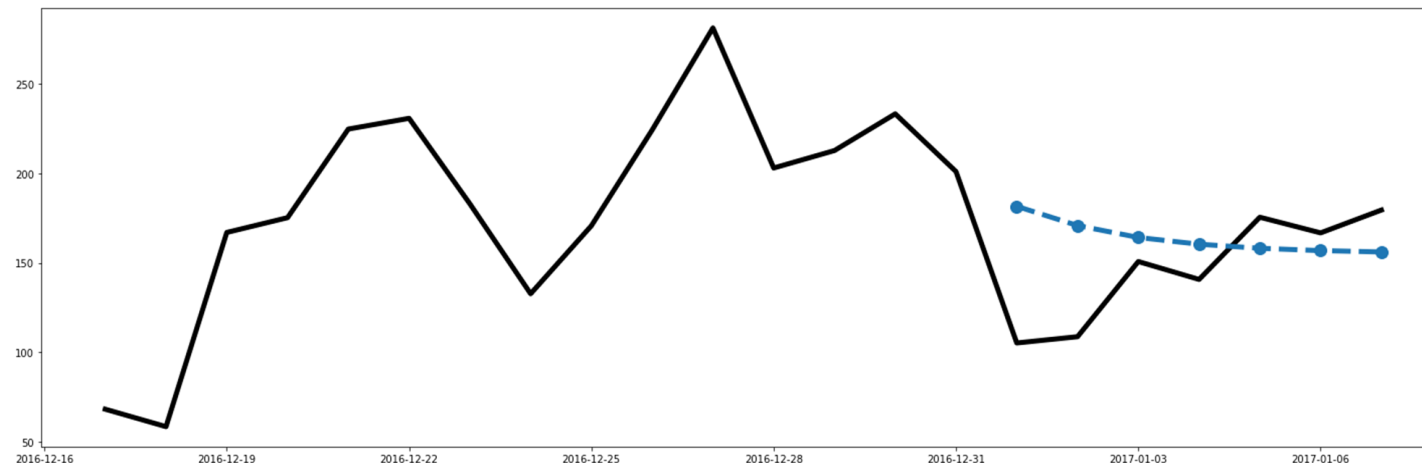


```
ini_forecast = pd.Timestamp("2017-01-01", freq="D")
ndays = 7

end_forecast = ini_forecast + ndays - 1
```

```
model_conf = ARIMA(seriecont[:ini_forecast-1], order=(2,1,2), freq="D")
model_fit = model_conf.fit()

forecast = model_fit.forecast(ndays)[0]
forecast = pd.DataFrame(forecast, columns=["forecast"],
                        index=seriecont[ini_forecast:end_forecast].index)
```



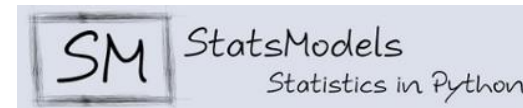
@mariamedp

@PiperLab_es



Modelos ARIMA base

Predecir día a día



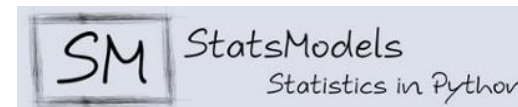
```
forecast_daily = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test - 1], order=(2,1,2), freq="D")
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=["forecast"],
                              index=seriecont[ini_forecast:end_forecast].index)
```



Modelos ARIMA base

Predecir día a día



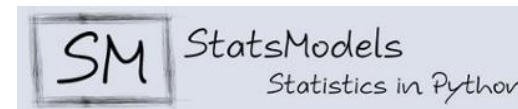
```
forecast_daily = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test - 1], order=(2,1,2), freq="D")
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=["forecast"],
                              index=seriecont[ini_forecast:end_forecast].index)
```



Modelos ARIMA base

Predecir día a día



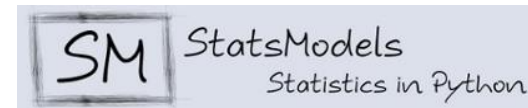
```
forecast_daily = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test - 1], order=(2,1,2), freq="D")
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=["forecast"],
                              index=seriecont[ini_forecast:end_forecast].index)
```



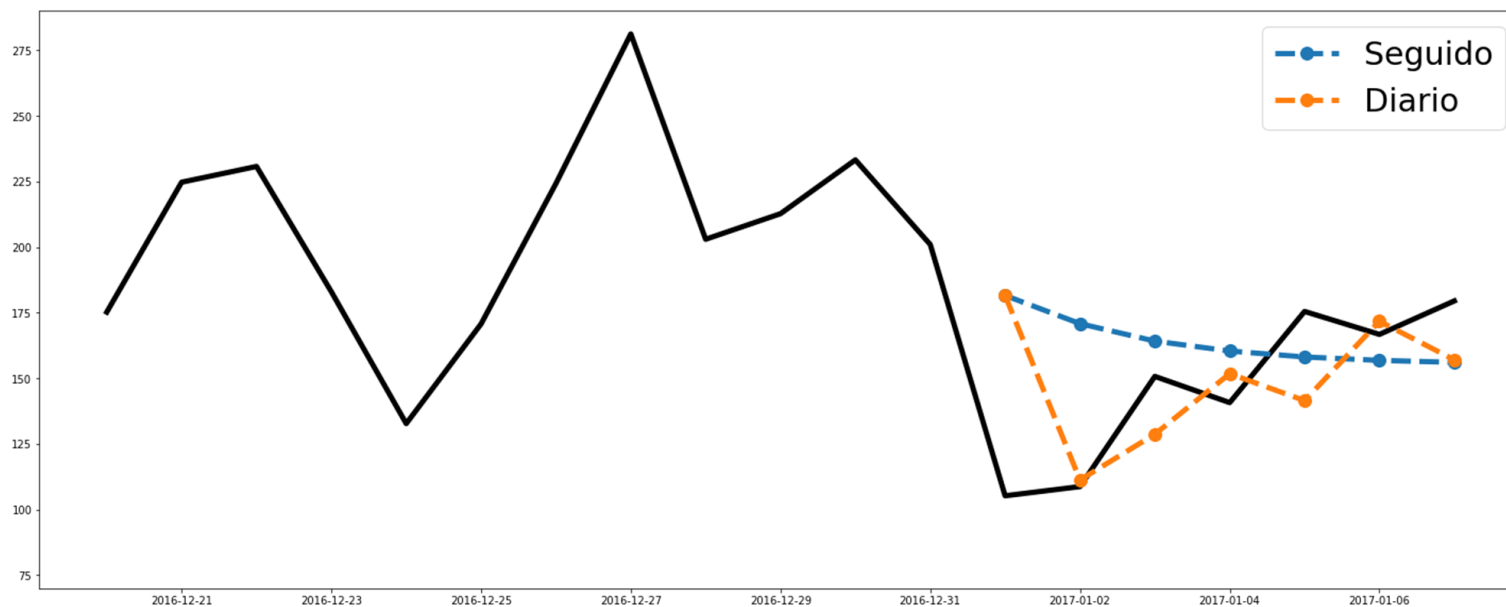

Modelos ARIMA base

Predecir día a día



```
forecast_daily = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test - 1], order=(2,1,2), freq="D")
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=["forecast"],
                              index=seriecont[ini_forecast:end_forecast].index)
```



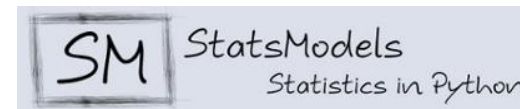
@mariamedp

@PiperLab_es



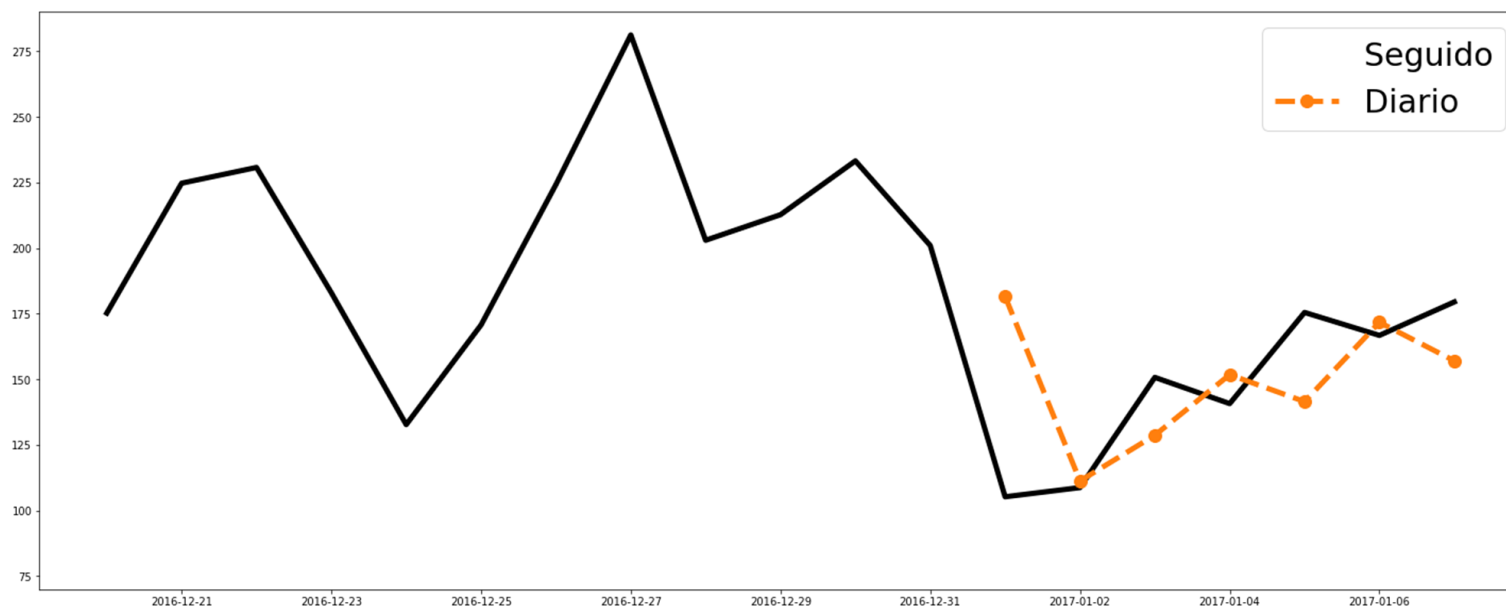
Modelos ARIMA base

Predecir día a día



```
forecast_daily = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test - 1], order=(2,1,2), freq="D")
    modelfit_day = modelconf_day.fit()
    forecast_daily.append(modelfit_day.forecast(1)[0])

forecast_daily = pd.DataFrame(forecast_daily, columns=["forecast"],
                              index=seriecont[ini_forecast:end_forecast].index)
```



@mariamedp

@PiperLab_es



Modelos con meteorología



Global Forecast System (GFS)

Datos = predicciones de contaminación

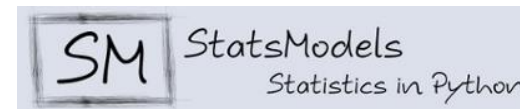
@mariamedp

@PiperLab_es



Modelos con meteorología

Predecir día a día



```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test+1])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                   index=seriecont[ini_forecast:end_forecast].index)
```

	var4571	var9886	var8446	var5201	var3497	var11758	var6383	var3485	var8470	var16165	var8518	var3494	var8434	var8422	var72
date															
2015-07-02	1536.03	212.7	55.0	1042.926667	5.731250	346.1	92266.2	6.925000	39.3	7987.500	37.9	10.3	45.4	47.4	35
2015-07-03	1559.85	212.4	59.5	1136.186667	7.340625	316.2	92378.7	8.133333	49.3	11725.000	63.9	9.5	43.4	59.5	40
2015-07-04	1572.43	212.6	53.3	916.513333	5.209375	310.3	92408.5	5.791667	47.1	7012.500	45.2	7.9	47.4	53.3	37
2015-07-05	1553.66	212.9	49.7	952.100000	5.243750	321.9	92323.7	6.866667	42.1	9000.000	35.3	8.5	34.7	49.7	37
2015-07-06	1559.65	212.0	39.6	1147.200000	4.612500	308.4	92270.4	6.208333	32.5	8750.025	27.6	7.4	32.1	39.6	30

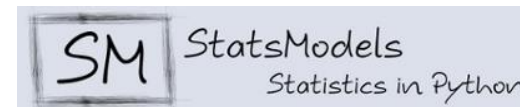
@mariamedp

@PiperLab_es



Modelos con meteorología

Predecir día a día



```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test+1])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                     index=seriecont[ini_forecast:end_forecast].index)
```

	var4571	var9886	var8446	var5201	var3497	var11758	var6383	var3485	var8470	var16165	var8518	var3494	var8434	var8422	var72
date															
2015-07-02	1536.03	212.7	55.0	1042.926667	5.731250	346.1	92266.2	6.925000	39.3	7987.500	37.9	10.3	45.4	47.4	35
2015-07-03	1559.85	212.4	59.5	1136.186667	7.340625	316.2	92378.7	8.133333	49.3	11725.000	63.9	9.5	43.4	59.5	40
2015-07-04	1572.43	212.6	53.3	916.513333	5.209375	310.3	92408.5	5.791667	47.1	7012.500	45.2	7.9	47.4	53.3	37
2015-07-05	1553.66	212.9	49.7	952.100000	5.243750	321.9	92323.7	6.866667	42.1	9000.000	35.3	8.5	34.7	49.7	37
2015-07-06	1559.65	212.0	39.6	1147.200000	4.612500	308.4	92270.4	6.208333	32.5	8750.025	27.6	7.4	32.1	39.6	30

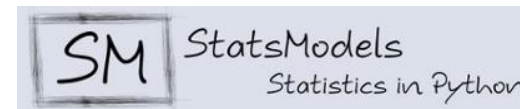
@mariamedp

@PiperLab_es



Modelos con meteorología

Predecir día a día



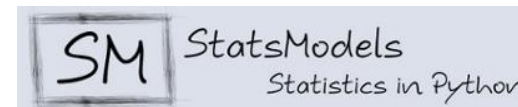
```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                   index=seriecont[ini_forecast:end_forecast].index)
```



Modelos con meteorología

Predecir día a día



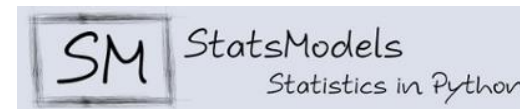
```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test+1])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                    index=seriecont[ini_forecast:end_forecast].index)
```



Modelos con meteorología

Predecir día a día



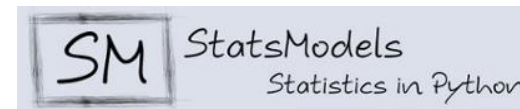
```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                    index=seriecont[ini_forecast:end_forecast].index)
```



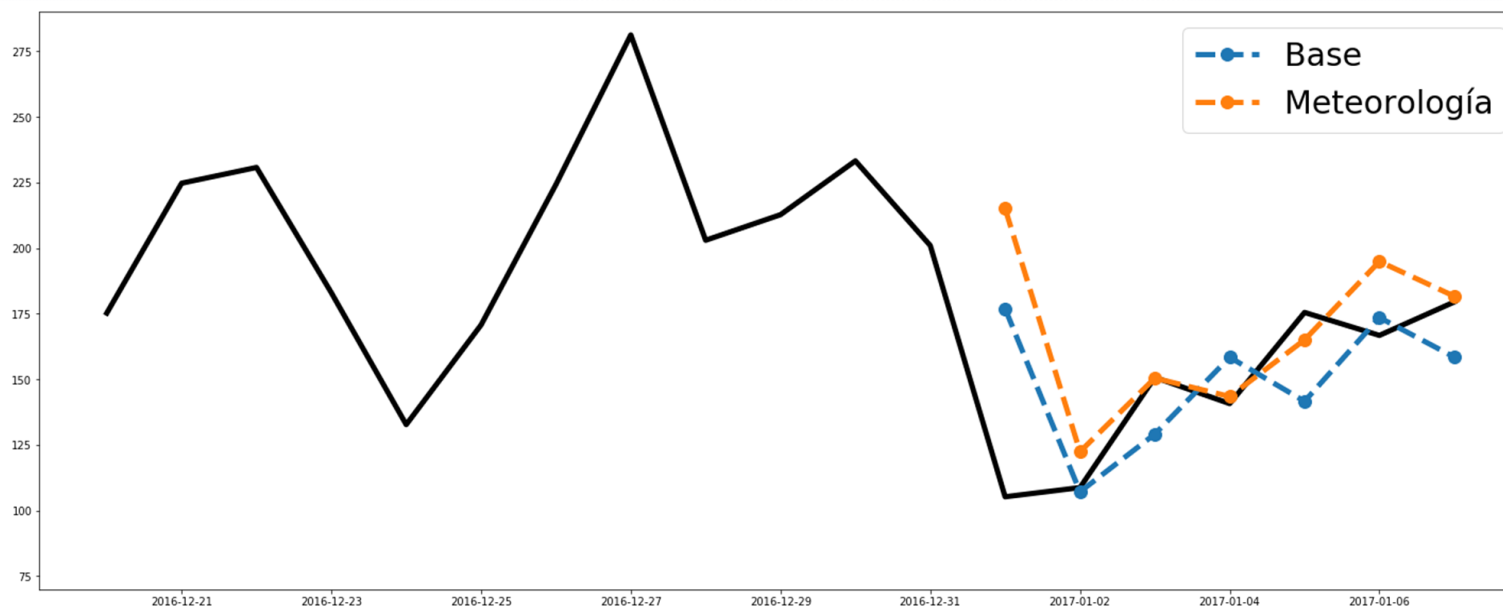

Modelos con meteorología

Predecir día a día



```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test+1])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                   index=seriecont[ini_forecast:end_forecast].index)
```



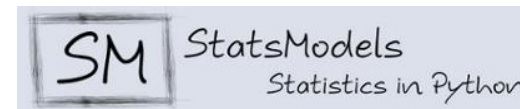
@mariamedp

@PiperLab_es



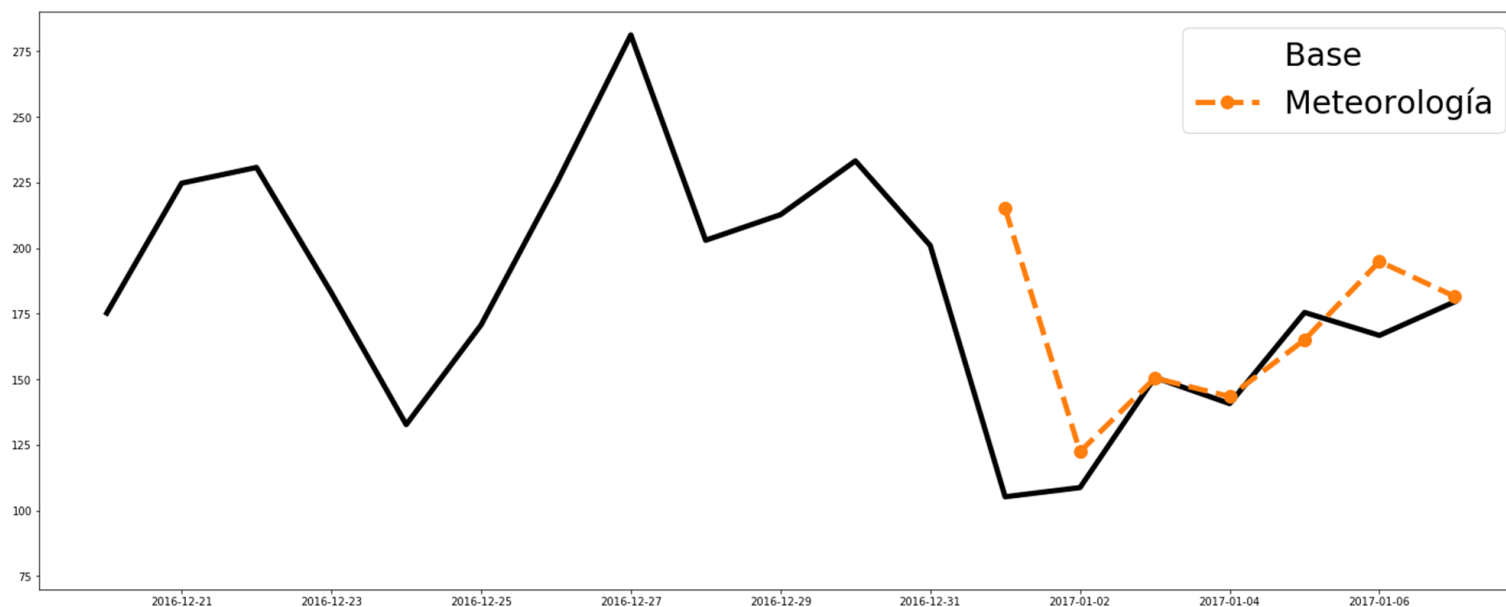
Modelos con meteorología

Predecir día a día



```
forecast_daily_meteo = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_meteo.append(modelfit_day.forecast(1, exog=meteo_pred[ini_test:ini_test+1])[0])

forecast_daily_meteo = pd.DataFrame(forecast_daily_meteo, columns=["forecast"],
                                   index=seriecont[ini_forecast:end_forecast].index)
```



@mariamedp

@PiperLab_es



Transformación de variables: PCA



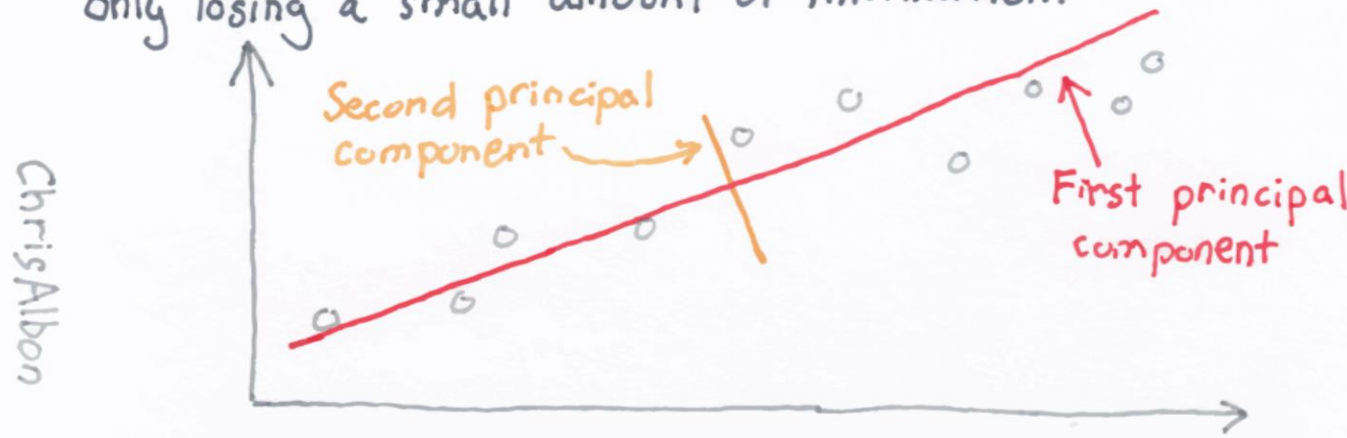


Transformación de variables: PCA

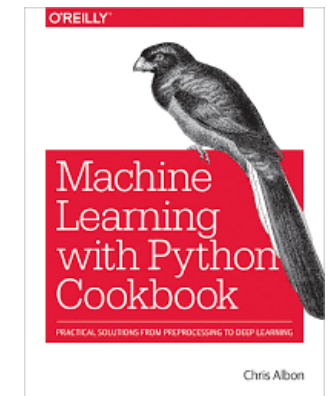
PCA

PRINCIPAL COMPONENT ANALYSIS

PCA projects the features onto the principal components. The motivation is to reduce the features dimensionality while only losing a small amount of information.



@chrisalbon



@mariamedp

@PiperLab_es

Aplicar PCA

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler().fit(meteo_real)

meteo_real_scaled = scaler.transform(meteo_real)
meteo_pred_scaled = scaler.transform(meteo_pred)

pca = PCA(n_components=10).fit(meteo_real_scaled)

meteo_real_pca = pca.transform(meteo_real_scaled)
meteo_pred_pca = pca.transform(meteo_pred_scaled)

meteo_real_pca = pd.DataFrame(meteo_real_pca, index=meteo_real.index)
meteo_pred_pca = pd.DataFrame(meteo_pred_pca, index=meteo_pred.index)
```



Aplicar PCA



```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler().fit(meteo_real)

meteo_real_scaled = scaler.transform(meteo_real)
meteo_pred_scaled = scaler.transform(meteo_pred)

pca = PCA(n_components=10).fit(meteo_real_scaled)

meteo_real_pca = pca.transform(meteo_real_scaled)
meteo_pred_pca = pca.transform(meteo_pred_scaled)

meteo_real_pca = pd.DataFrame(meteo_real_pca, index=meteo_real.index)
meteo_pred_pca = pd.DataFrame(meteo_pred_pca, index=meteo_pred.index)
```

Aplicar PCA



```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler().fit(meteo_real)

meteo_real_scaled = scaler.transform(meteo_real)
meteo_pred_scaled = scaler.transform(meteo_pred)

pca = PCA(n_components=10).fit(meteo_real_scaled)

meteo_real_pca = pca.transform(meteo_real_scaled)
meteo_pred_pca = pca.transform(meteo_pred_scaled)

meteo_real_pca = pd.DataFrame(meteo_real_pca, index=meteo_real.index)
meteo_pred_pca = pd.DataFrame(meteo_pred_pca, index=meteo_pred.index)
```




Aplicar PCA



```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler().fit(meteo_real)

meteo_real_scaled = scaler.transform(meteo_real)
meteo_pred_scaled = scaler.transform(meteo_pred)

pca = PCA(n_components=10).fit(meteo_real_scaled)

meteo_real_pca = pca.transform(meteo_real_scaled)
meteo_pred_pca = pca.transform(meteo_pred_scaled)

meteo_real_pca = pd.DataFrame(meteo_real_pca, index=meteo_real.index)
meteo_pred_pca = pd.DataFrame(meteo_pred_pca, index=meteo_pred.index)
```



	0	1	2	3	4	5	6	7	8	9
date										
2015-07-02	-0.504376	2.798512	-0.852335	-0.780081	-0.895056	-0.257762	-0.308637	0.076789	0.334893	-0.353400
2015-07-03	0.082084	3.036624	1.045996	-1.167916	-0.607475	-0.472422	0.669803	0.270863	-0.188384	-0.172179
2015-07-04	-1.604054	1.966525	0.330712	-0.998042	-0.176281	-0.312145	0.350273	0.609419	-0.170290	0.277479
2015-07-05	-1.451573	2.818816	-0.033557	-0.860587	-0.102741	-0.229093	0.357474	0.695917	-0.199327	-0.123499
2015-07-06	-1.815973	3.705881	-0.157315	-1.223369	-0.006324	-1.004604	0.717226	0.359217	-0.540903	-0.190084

@mariamedp

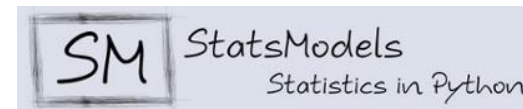
@PiperLab_es





Modelos con meteorología PCA

Predecir día a día



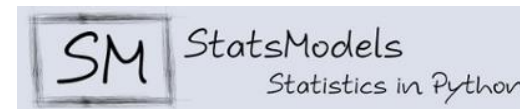
```
forecast_daily_pca = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real_pca[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_pca.append(modelfit_day.forecast(1, exog=meteo_pred_pca[ini_test:ini_test+1])[0])

forecast_daily_pca = pd.DataFrame(forecast_daily_pca, columns=["forecast"],
                                  index=seriecont[ini_forecast:end_forecast].index)
```



Modelos con meteorología PCA

Predecir día a día



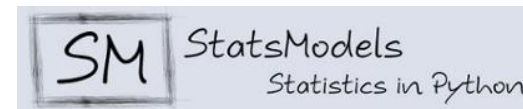
```
forecast_daily_pca = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real_pca[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_pca.append(modelfit_day.forecast(1, exog=meteo_pred_pca[ini_test:ini_test+1])[0])

forecast_daily_pca = pd.DataFrame(forecast_daily_pca, columns=["forecast"],
                                  index=seriecont[ini_forecast:end_forecast].index)
```



Modelos con meteorología PCA

Predecir día a día



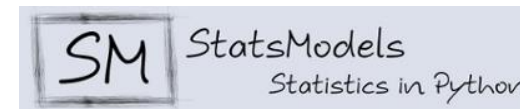
```
forecast_daily_pca = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real_pca[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_pca.append(modelfit_day.forecast(1, exog=meteo_pred_pca[ini_test:ini_test+1])[0])

forecast_daily_pca = pd.DataFrame(forecast_daily_pca, columns=["forecast"],
                                  index=seriecont[ini_forecast:end_forecast].index)
```



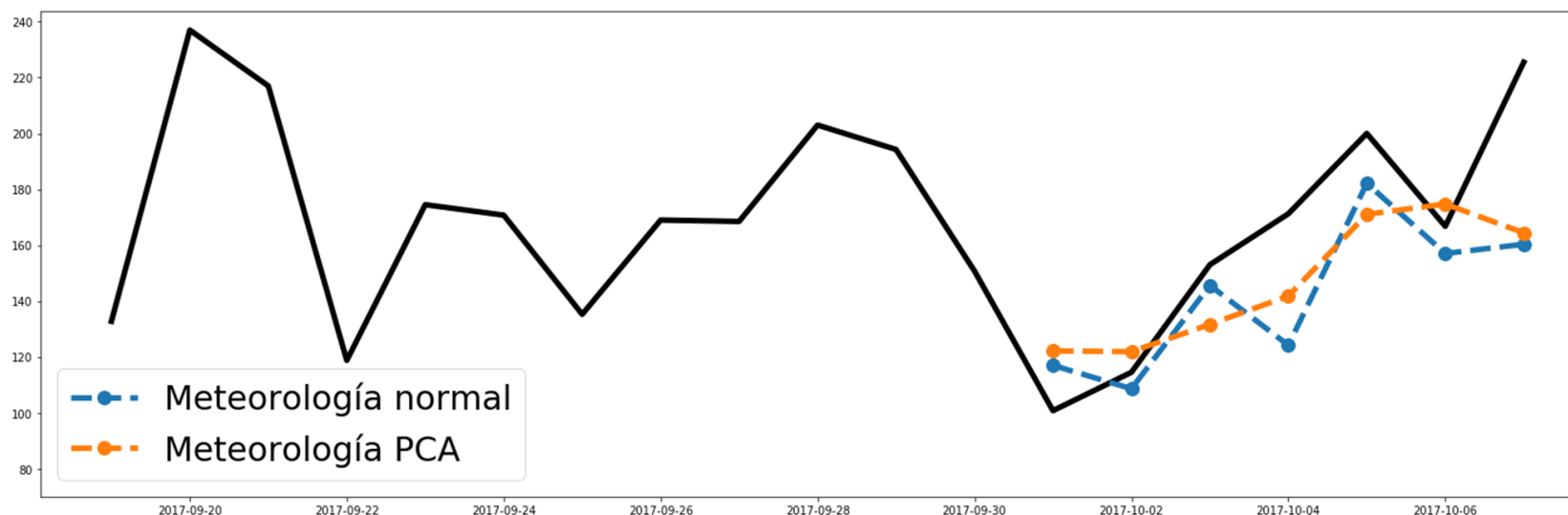
Modelos con meteorología PCA

Predecir día a día



```
forecast_daily_pca = []
for d in range(ndays):
    ini_test = ini_forecast + d
    modelconf_day = ARIMA(seriecont[:ini_test-1], order=(1,0,1), freq="D",
                          exog = meteo_real_pca[:ini_test-1])
    modelfit_day = modelconf_day.fit()
    forecast_daily_pca.append(modelfit_day.forecast(1, exog=meteo_pred_pca[ini_test:ini_test+1])[0])

forecast_daily_pca = pd.DataFrame(forecast_daily_pca, columns=["forecast"],
                                  index=seriecont[ini_forecast:end_forecast].index)
```



@mariamedp

@PiperLab_es



Proyecto

Iteraciones



Modelos base

Datos



Variables



Modelos



Evaluación



11%

Modelos con meteorología limitada



9%

Modelos con meteorología PCA





Proyecto

Resultados



Horizonte	Precisión (accuracy)	Detección picos (recall)
24 horas	87.98%	90.24%
48 horas	87.36%	73.17%
7 días	77.97%	51.22%

#PyConES18

¡Gracias!

@mariamedp



PIPERLAB