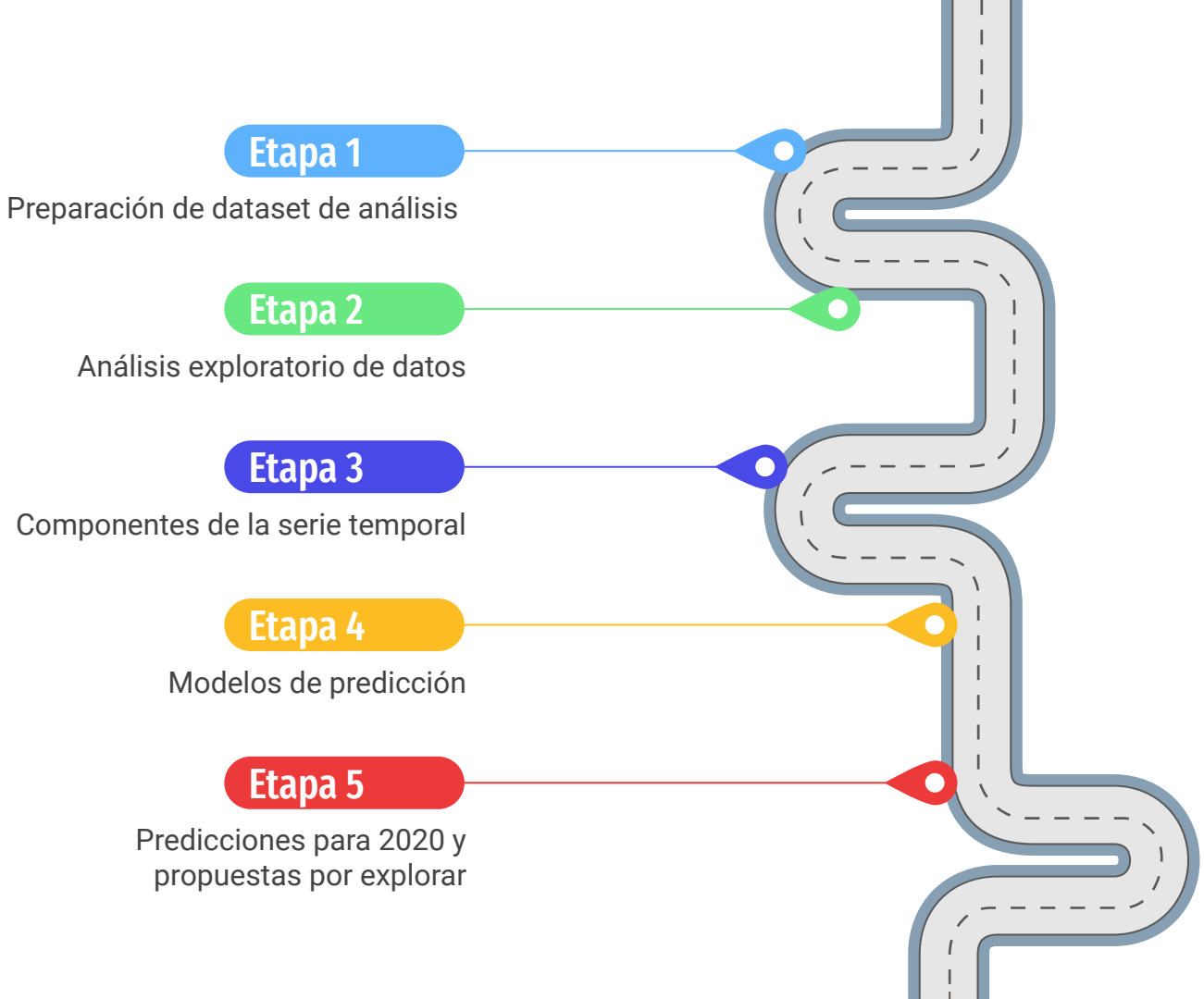
ACÁMICA

Carrera Data Science

Proyecto 03
Series de tiempo

Objetivo: implementar un modelo de machine learning para generar predicciones de flujo vehicular en autopistas de la ciudad de Buenos Aires, Argentina

Presentado por : Alexander Ortega
(Febrero 2021)


Etapa 1
Preparación de dataset de análisis
Etapa 2
Análisis exploratorio de datos
Etapa 3
Componentes de la serie temporal
Etapa 4
Modelos de predicción
Etapa 5
Predicciones para 2020 y propuestas por explorar

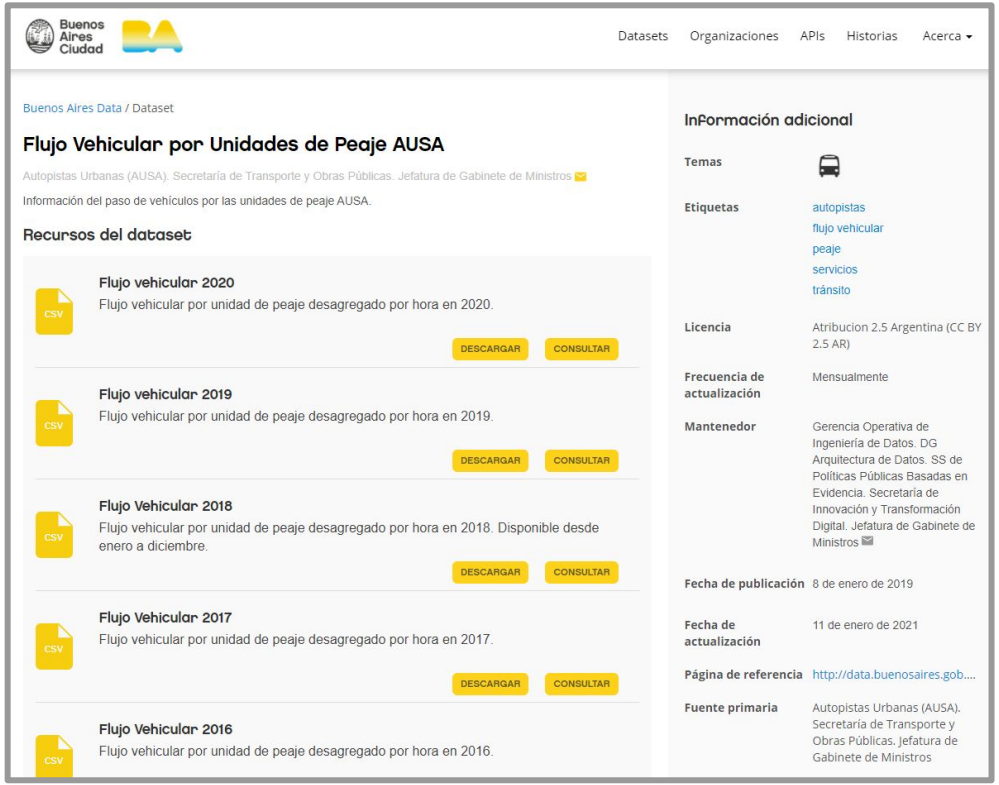
Etapa 1

Preparación del dataset de análisis

Step 1

Etapa 1

Recopilación y descarga de la información primaria



Step 2

Exploración y preparación del dataset base

Exploración de la información base

Etapa 1

```
data_2017.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372645 entries, 0 to 1372644
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   periodo         1372645 non-null  int64
 1   fecha           1372645 non-null  object
 2   hora_inicio     1372645 non-null  int64
 3   hora_fin        1372645 non-null  int64
 4   dia             1372645 non-null  object
 5   estacion        1372645 non-null  object
 6   sentido         1372645 non-null  object
 7   tipo_vehiculo   1372645 non-null  object
 8   forma_pago      1372645 non-null  object
 9   cantidad_pasos  1372645 non-null  int64
dtypes: int64(4), object(6)
memory usage: 104.7+ MB
```

```
data_2018.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1375908 entries, 0 to 1375907
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   periodo         1375908 non-null  int64
 1   fecha           1375908 non-null  object
 2   hora_inicio     1375908 non-null  int64
 3   hora_fin        1375908 non-null  int64
 4   dia             1375908 non-null  object
 5   estacion        1375908 non-null  object
 6   sentido         1375908 non-null  object
 7   tipo_vehiculo   1375908 non-null  object
 8   forma_pago      1375908 non-null  object
 9   cantidad_pasos  1375908 non-null  int64
dtypes: int64(4), object(6)
memory usage: 105.0+ MB
```

```
data_2019.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 812153 entries, 0 to 812152
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   periodo         812153 non-null  int64
 1   fecha           812153 non-null  object
 2   hora_inicio     812153 non-null  int64
 3   hora_fin        812153 non-null  int64
 4   dia             812153 non-null  object
 5   estacion        812153 non-null  object
 6   sentido         812153 non-null  object
 7   tipo_vehiculo   812153 non-null  object
 8   forma_pago      812153 non-null  object
 9   cantidad_pasos  812153 non-null  int64
dtypes: int64(4), object(6)
memory usage: 62.0+ MB
```

La información de sus atributos coincide (10 columnas en total)

La tipos de las variables no son los más útiles (fechas en específico)

Step 2

Etapa 1

Exploración y preparación del dataset base

Exploración de la información base

Los atributos sentido y tipo de vehículo coinciden

Los atributos forma de pago y estación deben homogenizarce

```
data_2017.head()
```

	periodo	fecha	hora_inicio	hora_fin	dia	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017	2017-01-01	0	1	Domingo	Alberdi	Centro	Liviano	NO COBRADO	25
1	2017	2017-01-01	0	1	Domingo	Alberdi	Centro	Liviano	TELEPASE	7
2	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	NO COBRADO	5
3	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	EFECTIVO	2
4	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	EFECTIVO	94

```
data_2018.head()
```

	periodo	fecha	hora_inicio	hora_fin	dia	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2018	2018-01-01	0	1	Lunes	Alberdi	Centro	Liviano	NO COBRADO	29
1	2018	2018-01-01	0	1	Lunes	Alberdi	Centro	Liviano	TELEPASE	9
2	2018	2018-01-01	1	2	Lunes	Alberdi	Centro	Liviano	NO COBRADO	73
3	2018	2018-01-01	1	2	Lunes	Alberdi	Centro	Liviano	TELEPASE	39
4	2018	2018-01-01	2	3	Lunes	Alberdi	Centro	Liviano	NO COBRADO	115

```
data_2019.head()
```

	periodo	fecha	hora_inicio	hora_fin	dia	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2019	2019-01-01	0	1	Martes	Alberti	Centro	Liviano	NO COBRADO	22
1	2019	2019-01-01	0	1	Martes	Alberti	Centro	Liviano	TELEPASE	6
2	2019	2019-01-01	0	1	Martes	Alberti	Provincia	Liviano	NO COBRADO	53
3	2019	2019-01-01	0	1	Martes	Alberti	Provincia	Liviano	TELEPASE	18
4	2019	2019-01-01	0	1	Martes	Avellaneda	Centro	Liviano	EFECTIVO	16

```
print(data_2019['estacion'].unique())
print(data_2018['estacion'].unique())
print(data_2017['estacion'].unique())
```

```
['Alberti' 'Avellaneda' 'Dellepiane' 'Illia' 'Retiro' 'Salguero'
 'Sarmiento' 'PDB']
['Alberdi' 'Avellaneda' 'Dellepiane Centro' 'Dellepiane Liniers' 'Illia'
 'Retiro' 'Salguero' 'Sarmiento']
['Alberdi' 'Avellaneda' 'Dellepiane Centro' 'Dellepiane Liniers' 'Illia'
 'Retiro' 'Salguero' 'Sarmiento']
```

```
print(data_2019['sentido'].unique())
print(data_2018['sentido'].unique())
print(data_2017['sentido'].unique())
```

```
['Centro' 'Provincia']
['Centro' 'Provincia']
['Centro' 'Provincia']
```

```
print(data_2019['tipo_vehiculo'].unique())
print(data_2018['tipo_vehiculo'].unique())
print(data_2017['tipo_vehiculo'].unique())
```

```
['Liviano' 'Pesado']
['Liviano' 'Pesado']
['Liviano' 'Pesado']
```

```
print(data_2019['forma_pago'].unique())
print(data_2018['forma_pago'].unique())
print(data_2017['forma_pago'].unique())
```

```
['NO COBRADO' 'TELEPASE' 'EFECTIVO' 'EXENTO' 'INFRACCION'
 'T. DISCAPACIDAD']
['NO COBRADO' 'TELEPASE' 'EFECTIVO' 'EXENTO' 'TARJETA DISCAPACIDAD'
 'INFRACCION']
['NO COBRADO' 'TELEPASE' 'EFECTIVO' 'TARJETA DISCAPACIDAD' 'EXENTO'
 'INFRACCION' 'MONEDERO' 'Otros']
```

Step 2

Exploración y preparación del dataset base

Consolidación del dataset base

Etapa 1

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3560706 entries, 0 to 812152
Data columns (total 10 columns):
 #   Column          Dtype
---  ------          -----
 0   periodo         int64
 1   fecha           object
 2   hora_inicio     int64
 3   hora_fin        int64
 4   dia             object
 5   estacion        object
 6   sentido         object
 7   tipo_vehiculo   object
 8   forma_pago      object
 9   cantidad_pasos  int64
dtypes: int64(4), object(6)
memory usage: 298.8+ MB
```

```
dataset = pd.concat([data_2017, data_2018, data_2019])
dataset
```

	periodo	fecha	hora_inicio	hora_fin	dia	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017	2017-01-01	0	1	Domingo	Alberdi	Centro	Liviano	NO COBRADO	25
1	2017	2017-01-01	0	1	Domingo	Alberdi	Centro	Liviano	TELEPASE	7
2	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	NO COBRADO	5
3	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	EFECTIVO	2
4	2017	2017-01-01	1	2	Domingo	Alberdi	Centro	Liviano	EFECTIVO	94
...	...	...	...	...	...	...	...	...	...	...
812148	2019	2019-12-31	23	0	Martes	Salguero	Provincia	Liviano	NO COBRADO	7
812149	2019	2019-12-31	23	0	Martes	Salguero	Provincia	Liviano	TELEPASE	4
812150	2019	2019-12-31	23	0	Martes	Salguero	Provincia	Pesado	NO COBRADO	1
812151	2019	2019-12-31	23	0	Martes	Sarmiento	Provincia	Liviano	NO COBRADO	7
812152	2019	2019-12-31	23	0	Martes	Sarmiento	Provincia	Liviano	TELEPASE	7

3560706 rows × 10 columns

```
dataset.isnull().sum()
```

```
periodo           0
fecha             0
hora_inicio       0
hora_fin          0
dia               0
estacion          0
sentido           0
tipo_vehiculo     0
forma_pago        0
cantidad_pasos    0
dtype: int64
```

Los atributos sentido y tipo de vehículo coinciden

Los atributos forma de pago y estación deben homogenizarce

Los atributos de fecha se procesarán de manera particular

Step 3

Etapa 1

Procesamiento de atributos generales

Estación

```
dataset.estacion.unique()

array(['Alberdi', 'Avellaneda', 'Dellepiane Centro', 'Dellepiane Liniers',
       'Illia', 'Retiro', 'Salguero', 'Sarmiento', 'Alberti',
       'Dellepiane', 'PDB'], dtype=object)
```

```
dataset.estacion.replace({'Dellepiane Centro': 'Dellepiane', 'Dellepiane Liniers': 'Dellepiane',
                          'Alberdi': 'Alberti'}, inplace=True)
```

```
dataset.estacion.unique()

array(['Alberti', 'Avellaneda', 'Dellepiane', 'Illia', 'Retiro',
       'Salguero', 'Sarmiento', 'PDB'], dtype=object)
```

Step 3

Etapa 1

Procesamiento de atributos generales

Forma de pago

```
dataset.forma_pago.unique()

array(['NO COBRADO', 'TELEPASE', 'EFECTIVO', 'TARJETA DISCAPACIDAD',
       'EXENTO', 'INFRACCION', 'MONEDERO', 'Otros', 'T. DISCAPACIDAD'],
      dtype=object)
```

```
dataset.forma_pago.replace({'TARJETA DISCAPACIDAD': 'T. DISCAPACIDAD'}, inplace=True)
```

```
dataset.forma_pago.value_counts()

TELEPASE            1027861
EFECTIVO            1020176
EXENTO               464708
INFRACCION           405034
NO COBRADO           380450
T. DISCAPACIDAD      262455
MONEDERO                 14
Otros                     8
Name: forma_pago, dtype: int64
```

```
indx_drop=dataset[(dataset.forma_pago=='MONEDERO') | (dataset.forma_pago=='Otros')].index
dataset.drop(indx_drop, inplace=True)
```

```
dataset.forma_pago.unique()

array(['NO COBRADO', 'TELEPASE', 'EFECTIVO', 'T. DISCAPACIDAD', 'EXENTO',
       'INFRACCION'], dtype=object)
```

Step 4

Etapa 1

Procesamiento de atributos

Procesamiento de atributos relacionados a fechas

```
dataset.hora_fin.replace({0:24}, inplace=True)
```

```
dataset.fecha = pd.to_datetime(dataset.fecha)
```

```
dataset['mes'] = dataset['fecha'].dt.month
```

```
dataset['dia_num'] = dataset['fecha'].dt.weekday
```

```
dataset['fecha_comp'] = dataset.fecha + pd.to_timedelta(dataset.hora_fin, unit = 'h')
```

```
dataset['fecha_num'] = dataset.fecha.map(datetime.datetime.toordinal)
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3560706 entries, 0 to 812152
Data columns (total 10 columns):
 #   Column          Dtype
---  ------          -----
 0   periodo         int64
 1   fecha           object
 2   hora_inicio     int64
 3   hora_fin        int64
 4   dia             object
 5   estacion        object
 6   sentido         object
 7   tipo_vehiculo   object
 8   forma_pago      object
 9   cantidad_pasos  int64
dtypes: int64(4), object(6)
memory usage: 298.8+ MB
```

```
dataset = pd.concat([data_2017, data_2018, data_2019])
dataset
```

	periodo	fecha	hora_inicio	hora_fin	dia
0	2017	2017-01-01	0	1	Domingo
1	2017	2017-01-01	0	1	Domingo
2	2017	2017-01-01	1	2	Domingo
3	2017	2017-01-01	1	2	Domingo
4	2017	2017-01-01	1	2	Domingo
...	...	...	...	...	...
812148	2019	2019-12-31	23	0	Martes
812149	2019	2019-12-31	23	0	Martes
812150	2019	2019-12-31	23	0	Martes
812151	2019	2019-12-31	23	0	Martes
812152	2019	2019-12-31	23	0	Martes

3560706 rows × 10 columns

Step 4

Etapa 1

Procesamiento de atributos

Procesamiento de atributos relacionados a fechas

```
dataset.hora_fin.replace({0:24}, inplace=True)
```

```
dataset.fecha = pd.to_datetime(dataset.fecha)
```

```
dataset['mes'] = dataset['fecha'].dt.month
```

```
dataset['dia_num'] = dataset['fecha'].dt.weekday
```

```
dataset['fecha_comp'] = dataset.fecha + pd.to_timedelta(dataset.hora_fin, unit = 'h')
```

```
dataset['fecha_num'] = dataset.fecha.map(datetime.datetime.toordinal)
```

fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num
2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6
2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6
2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1
2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1

```
Column          Dtype
------          -----
fecha_comp      datetime64[ns]
fecha           datetime64[ns]
fecha_num       int64
periodo         int64
mes             int64
hora_inicio     int64
hora_fin        int64
dia             object
dia_num         int64
```

Step 5

Dataset base para el analisis

Etapa 1

```
dataset
```

	fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	25
1	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	TELEPASE	7
2	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	5
3	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	2
4	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	94
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
812148	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	NO COBRADO	7
812149	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	TELEPASE	4
812150	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Pesado	NO COBRADO	1
812151	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	NO COBRADO	7
812152	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	TELEPASE	7

3560648 rows × 14 columns

Step 5

Etapa 1

Dataset base para el analisis

```
 #   Column          Dtype
---  ------          -----
 0   fecha_comp      datetime64[ns]
 1   fecha           datetime64[ns]
 2   periodo         int64
 3   mes             int64
 4   hora_inicio     int64
 5   hora_fin        int64
 6   dia             object
 7   dia_num         int64
 8   estacion        object
 9   sentido         object
 10  tipo_vehiculo   object
 11  forma_pago      object
 12  cantidad_pasos  int64
```

```
dataset.estacion.unique()

array(['Alberti', 'Avellaneda', 'Dellepiane', 'Illia', 'Retiro',
       'Salguero', 'Sarmiento', 'PDB'], dtype=object)
```

dataset

	fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	25
1	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	TELEPASE	7
2	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	5

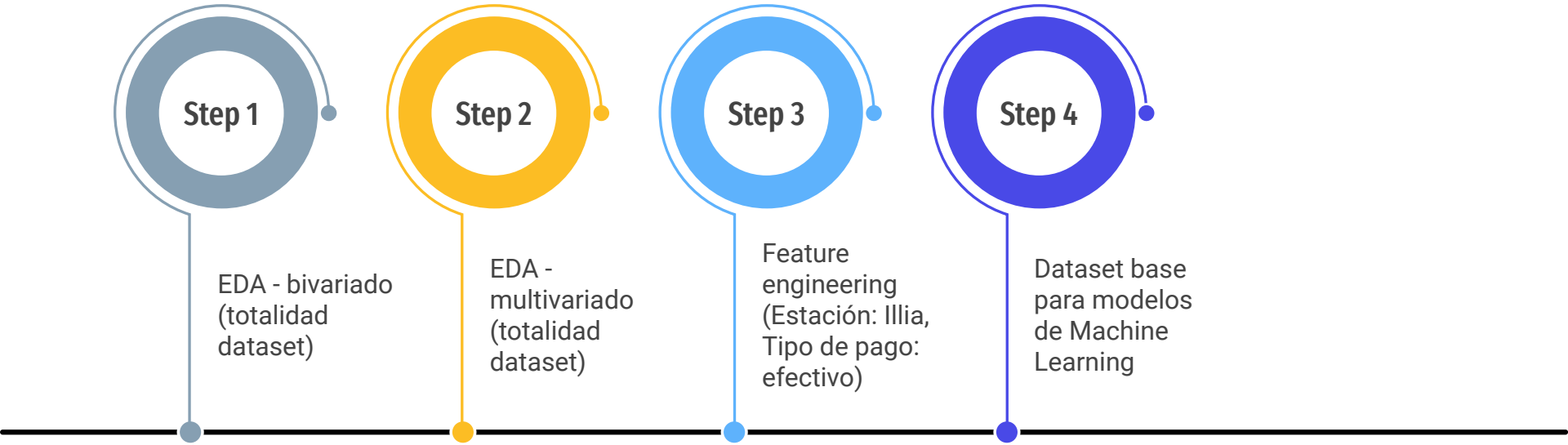
```
dataset.tipo_vehiculo.unique()

array(['Liviano', 'Pesado'], dtype=object)
```

```
dataset.forma_pago.unique()

array(['NO COBRADO', 'TELEPASE', 'EFECTIVO', 'T. DISCAPACIDAD', 'EXENTO',
       'INFRACCION'], dtype=object)
```

```
dataset.sentido.unique()

array(['Centro', 'Provincia'], dtype=object)
```

Etapa 2

Análisis exploratorio de datos



Step 1

EDA - bivariado (totalidad dataset)

Análisis exploratorio en términos de cada una de las variables del dataset

Etapa 2

```
dataset
```

	fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	25
1	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	TELEPASE	7
2	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	5
3	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	2
4	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	94
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
812148	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	NO COBRADO	7
812149	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	TELEPASE	4
812150	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Pesado	NO COBRADO	1
812151	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	NO COBRADO	7
812152	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	TELEPASE	7

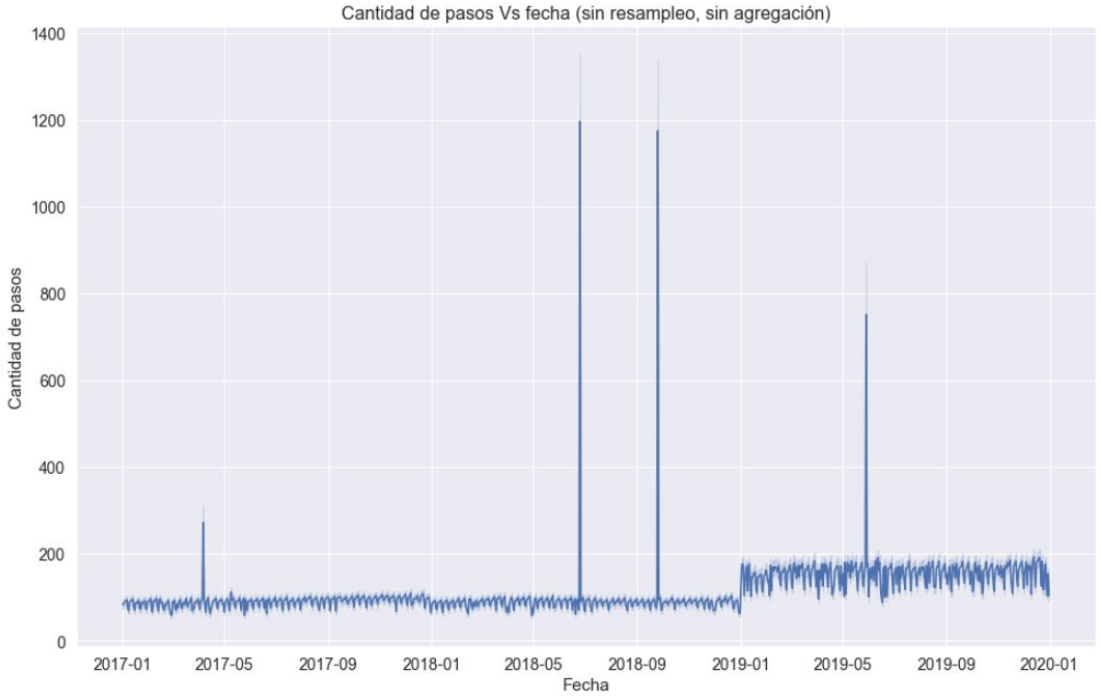
3560648 rows × 14 columns

Step 1

Etapa 2

EDA - bivariado (totalidad dataset)

Cantidad de pasos Vs fecha (sin resampleo, sin agregación)




```
dataset[['cantidad_pasos']].describe()
```

	cantidad_pasos
count	3.560648e+06
mean	1.052153e+02
std	3.206296e+02
min	1.000000e+00
25%	2.000000e+00
50%	8.000000e+00
75%	3.400000e+01
max	6.677000e+03

```
dataset[['periodo', 'cantidad_pasos']].groupby(['periodo']).describe()
```

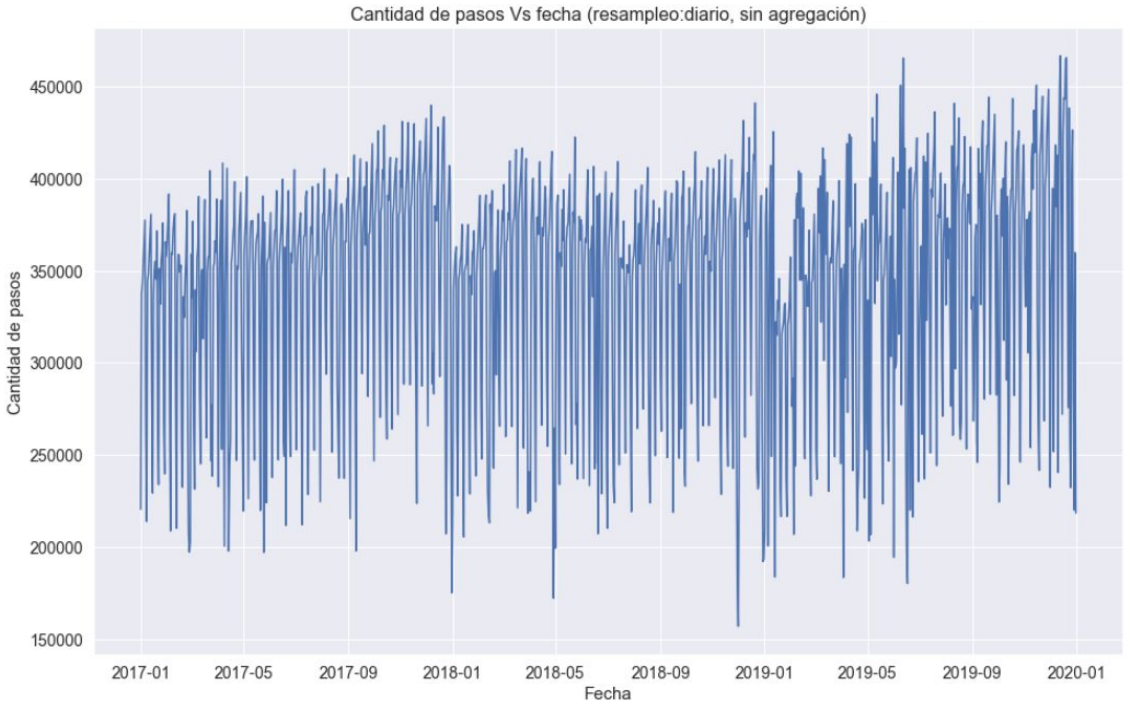
	cantidad_pasos							
periodo	count	mean	std	min	25%	50%	75%	max
2017	1372623.0	90.945185	301.640381	1.0	2.0	7.0	28.0	4271.0
2018	1375886.0	89.463204	291.091089	1.0	2.0	6.0	26.0	5722.0
2019	812139.0	156.020158	387.351812	1.0	4.0	15.0	72.0	6677.0

Step 1

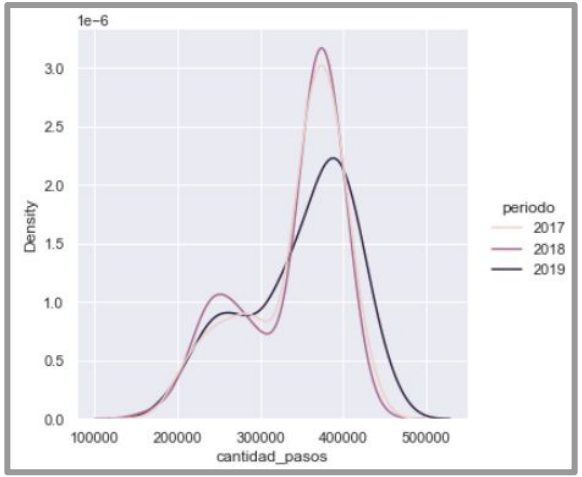
EDA - bivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: diario, sin agregación)

Etapa 2



	cantidad_pasos							
	count	mean	std	min	25%	50%	75%	max
periodo								
2017	365.0	342009.457534	59585.887708	174988.0	299491.0	359073.0	383828.0	439981.0
2018	365.0	337236.082192	60880.603983	156852.0	287589.0	361381.0	381546.0	441305.0
2019	365.0	347150.835616	66727.276771	180069.0	298206.0	363539.0	397094.0	466887.0

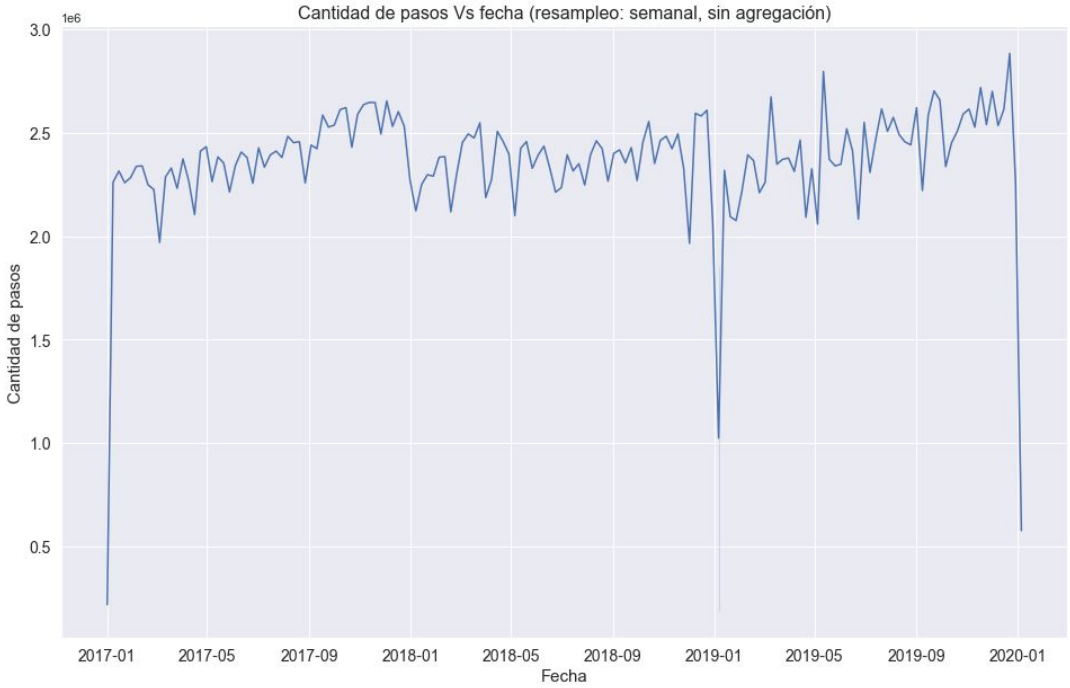


Step 1

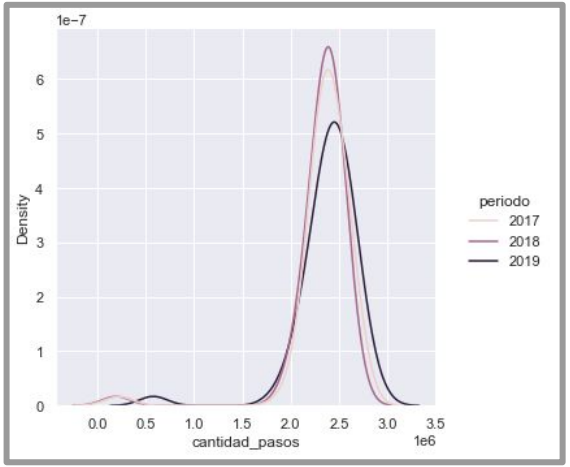
EDA - bivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: semanal, sin agregación)

Etapa 2



	cantidad_pasos							
	count	mean	std	min	25%	50%	75%	max
periodo								
2017	53.0	2.355348e+06	332977.628562	220442.0	2271255.0	2382623.0	2492836.0	2653142.0
2018	53.0	2.322475e+06	328023.094167	193355.0	2273366.0	2391074.0	2456470.0	2608145.0
2019	53.0	2.390756e+06	326434.524579	578292.0	2311791.0	2440537.0	2573575.0	2882347.0

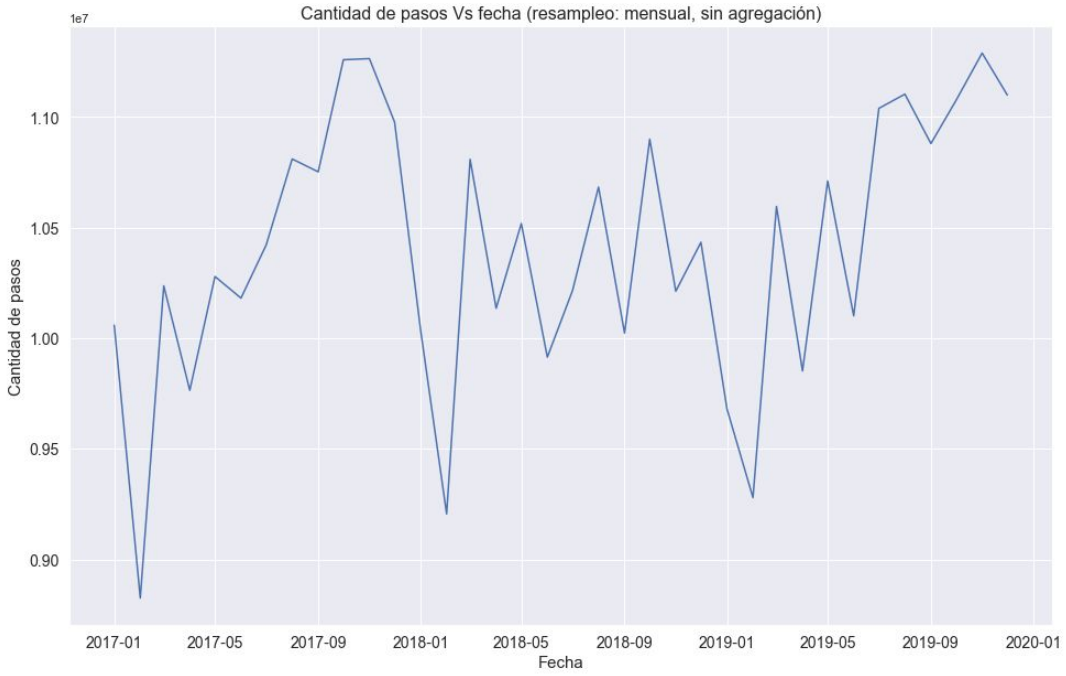


Step 1

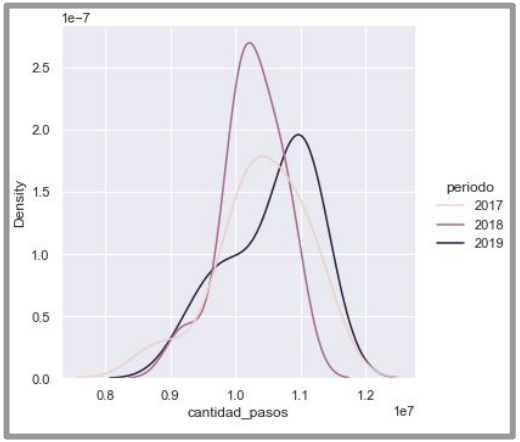
Etapa 2

EDA - bivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: mensual, sin agregación)



	cantidad_pasos							
	count	mean	std	min	25%	50%	75%	max
periodo								
2017	12.0	1.040279e+07	687482.006222	8825599.0	10151021.25	10350436.0	10852375.25	11264358.0
2018	12.0	1.025760e+07	461518.467173	9206000.0	10033958.75	10214180.5	10559969.75	10900036.0
2019	12.0	1.055917e+07	664972.146859	9279665.0	10039269.00	10795608.0	11081806.50	11289355.0



Step 1

Etapa 2

EDA - bivariado (totalidad dataset)

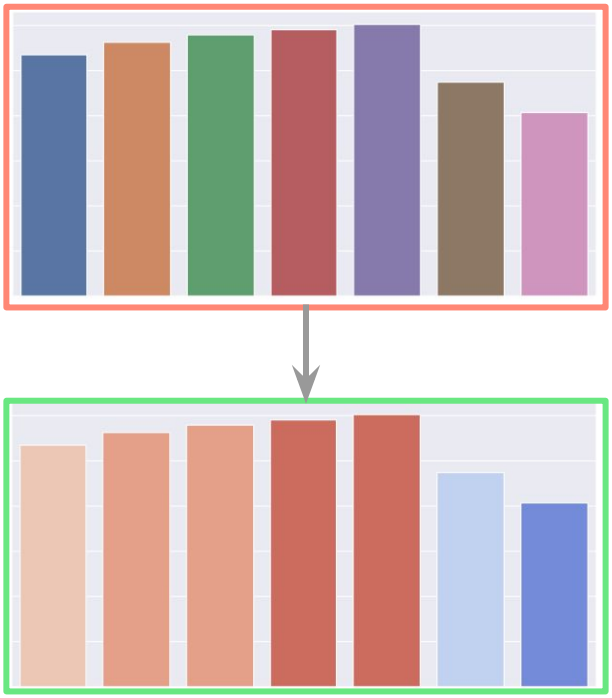
Bonus: función de normalizacion min max para la paleta de color

Bonus: función que nos permite tener una paleta normalizada

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

```python
# Función que nos permite definir la relación de color con base en el minimo
# y el maximo de un conjunto de datos y una paleta de color elegida
def colors_from_values(values, palette_name):
    # normalize the values to range [0, 1]
    normalized = (values - min(values)) / (max(values) - min(values))
    # convert to indices
    indices = np.round(normalized * (len(values) - 1)).astype(np.int32)
    # use the indices to get the colors
    palette = sns.color_palette(palette_name, len(values))
    return np.array(palette).take(indices, axis=0)
```

```python
palette=colors_from_values(dataset_analisis['cantidad_pasos'], "coolwarm"))
```

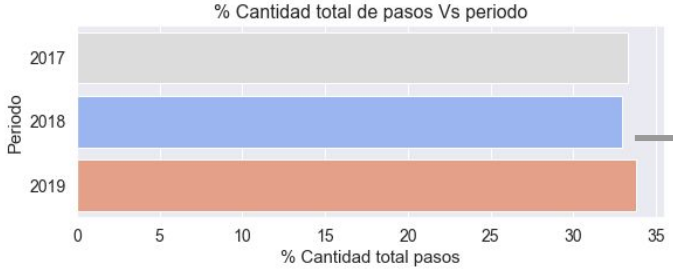


Step 1

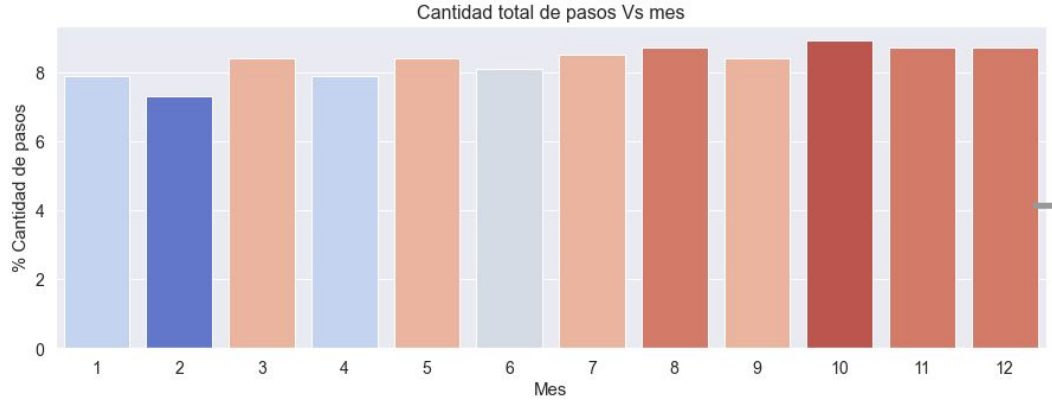
Etapa 2

EDA - bivariado (totalidad dataset)

% Cantidad total de pasos Vs periodo - mes (intervalo 2017 - 2019)



periodo	cantidad_pasos
2017	33.3
2018	32.9
2019	33.8



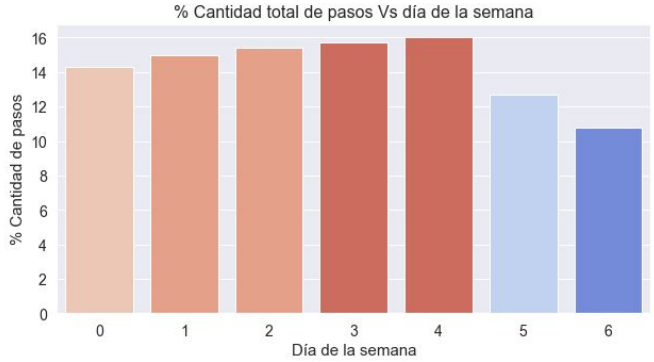
mes	cantidad_pasos
1	7.9
2	7.3
3	8.4
4	7.9
5	8.4
6	8.1
7	8.5
8	8.7
9	8.4
10	8.9
11	8.7
12	8.7

Step 1

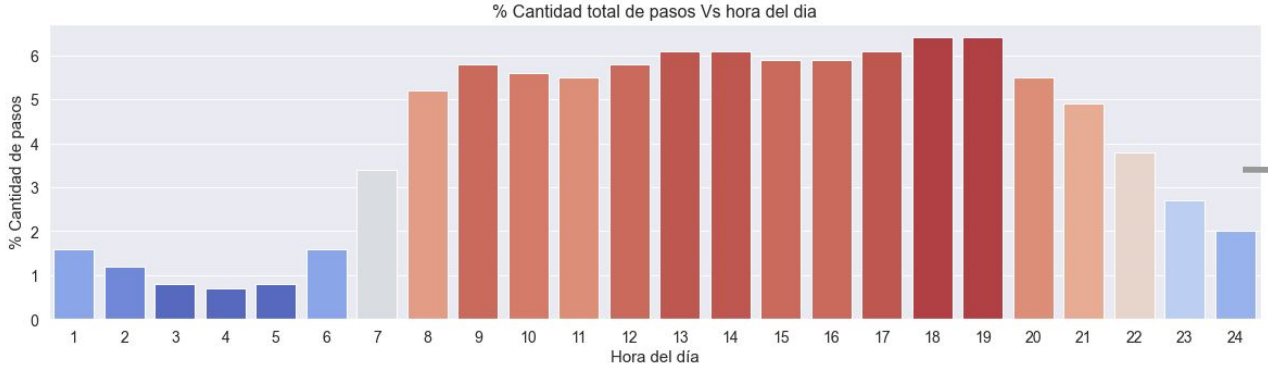
Etapa 2

EDA - bivariado (totalidad dataset)

% Cantidad total de pasos Vs día semana- hora día (intervalo 2017 - 2019)



dia_num	cantidad_pasos
0	14.3
1	15.0
2	15.4
3	15.7
4	16.0
5	12.7
6	10.8



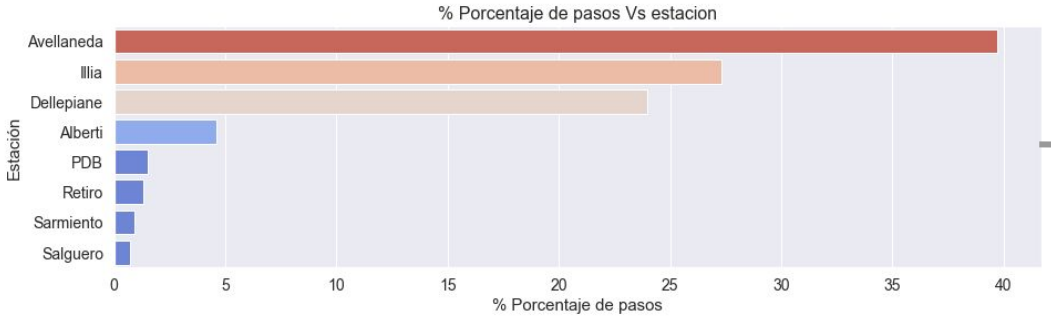
hora_fin	cantidad_pasos
1	1.6
2	1.2
3	0.8
4	0.7
5	0.8
6	1.6
7	3.4
8	5.2
9	5.8
10	5.6
11	5.5
12	5.8
13	6.1
14	6.1
15	5.9
16	5.9
17	6.1
18	6.4
19	6.4
20	5.5
21	4.9
22	3.8
23	2.7
24	2.0

Step 1

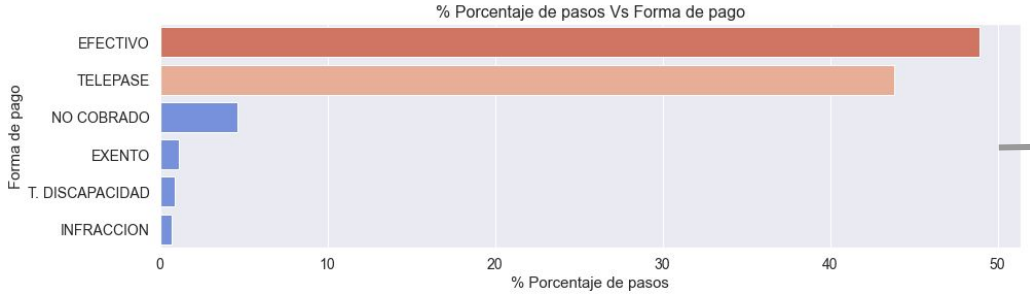
Etapa 2

EDA - bivariado (totalidad dataset)

% Cantidad total de pasos Vs estación - forma de pago (intervalo 2017 - 2019)



estacion	cantidad_pasos
Avellaneda	39.7
Illia	27.3
Dellepiane	24.0
Alberti	4.6
PDB	1.5
Retiro	1.3
Sarmiento	0.9
Salguero	0.7



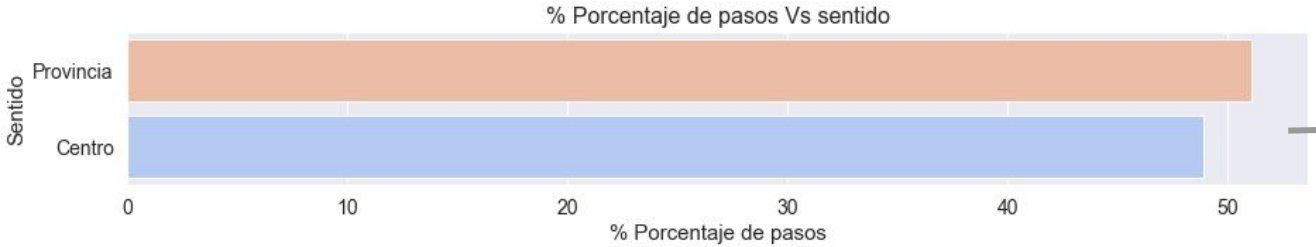
forma_pago	cantidad_pasos
EFECTIVO	48.9
TELEPASE	43.8
NO COBRADO	4.6
EXENTO	1.1
T. DISCAPACIDAD	0.9
INFRACCION	0.7

Step 1

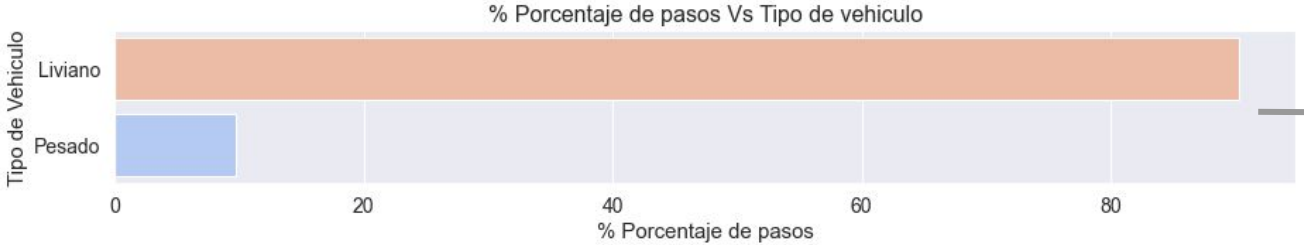
Etapa 2

EDA - bivariado (totalidad dataset)

% Cantidad total de pasos Vs sentido- tipo de vehículo (intervalo 2017 - 2019)



sentido	cantidad_pasos
Provincia	51.1
Centro	48.9



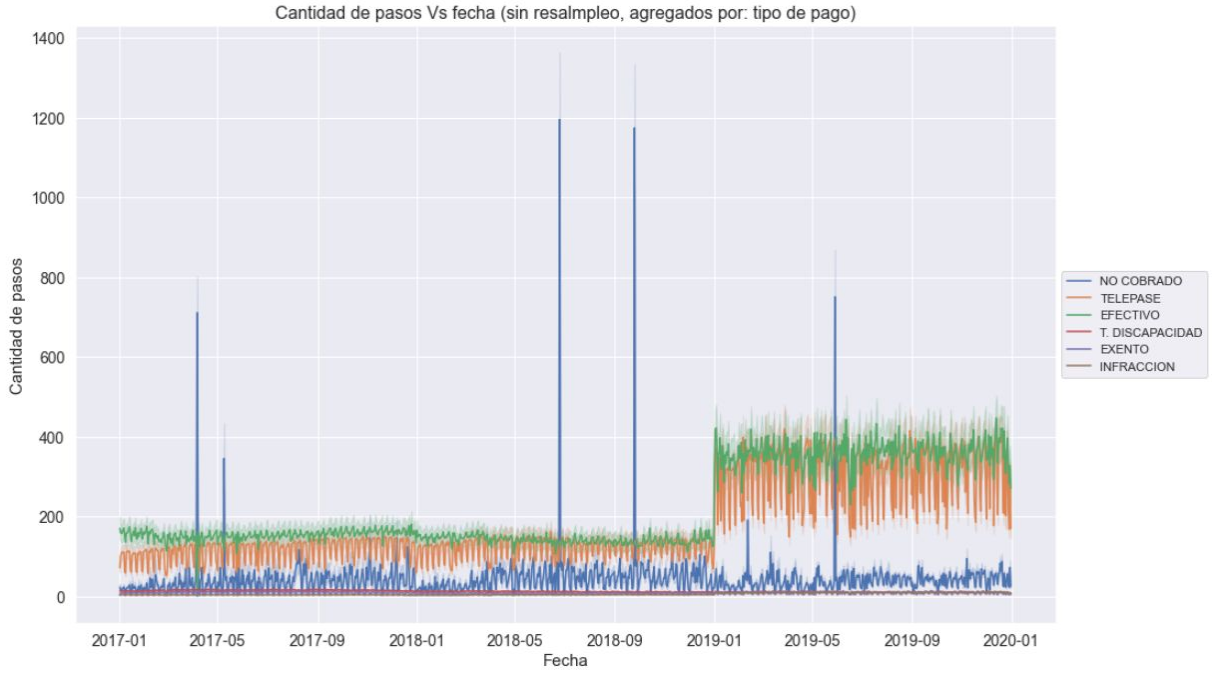
tipo_vehiculo	cantidad_pasos
Liviano	90.3
Pesado	9.7

Step 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs fecha (sin resampleo, agregado: tipo de pago)

Etapa 2



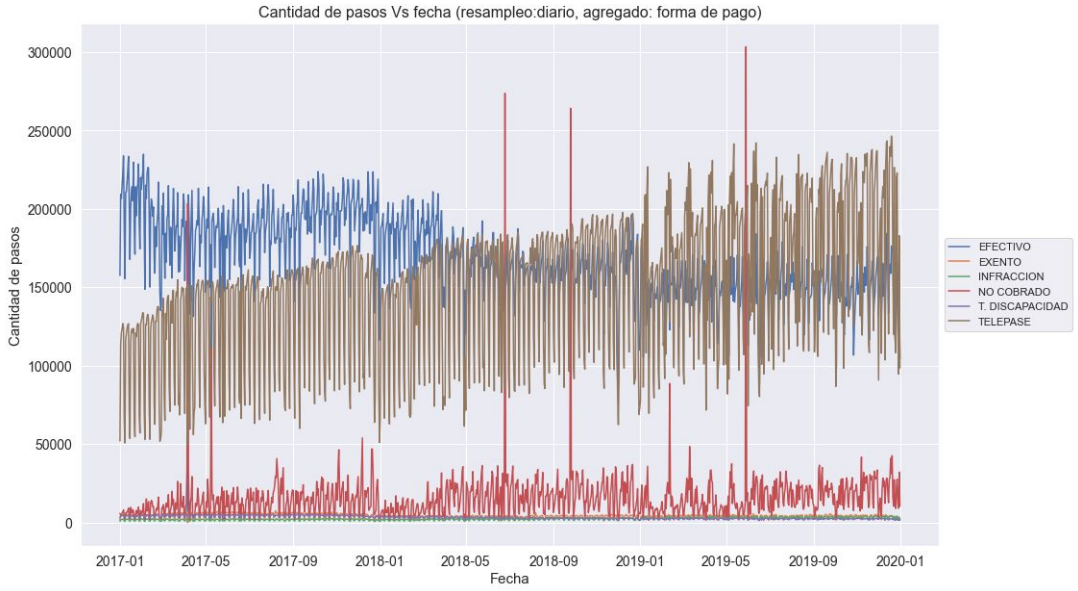
		cantidad_pasos							
		count	mean	std	min	25%	50%	75%	max
forma_pago	periodo								
EFECTIVO	2017	447203.0	155.416976	412.790403	1.0	3.0	10.0	56.0	3315.0
	2018	423622.0	142.163530	361.798749	1.0	2.0	8.0	53.0	2910.0
	2019	149338.0	359.060427	532.030328	1.0	16.0	69.0	517.0	2722.0
EXENTO	2017	169889.0	9.874077	15.736986	1.0	2.0	5.0	10.0	856.0
	2018	162100.0	7.714510	11.146307	1.0	2.0	4.0	9.0	709.0
	2019	132711.0	9.876499	13.031037	1.0	3.0	6.0	12.0	736.0
INFRACCION	2017	140275.0	4.487970	5.632329	1.0	1.0	2.0	5.0	57.0
	2018	153327.0	4.697777	6.132348	1.0	1.0	2.0	5.0	52.0
	2019	111426.0	9.858184	11.704023	1.0	2.0	5.0	13.0	84.0
NO COBRADO	2017	117730.0	41.159178	182.221584	1.0	1.0	5.0	22.0	4271.0
	2018	121439.0	50.004760	197.469703	1.0	1.0	5.0	24.0	5722.0
	2019	141280.0	44.363555	132.717882	1.0	3.0	12.0	44.0	6677.0
T. DISCAPACIDAD	2017	100106.0	15.634418	20.844572	1.0	1.0	6.0	22.0	152.0
	2018	86347.0	12.294336	14.793085	1.0	1.0	6.0	18.0	104.0
	2019	76000.0	11.074882	12.560349	1.0	2.0	6.0	16.0	82.0
TELEPASE	2017	397420.0	117.288254	315.386456	1.0	4.0	13.0	50.0	3335.0
	2018	429051.0	125.305957	347.252896	1.0	3.0	12.0	50.0	3656.0
	2019	201384.0	315.666274	532.871902	1.0	20.0	86.0	320.0	4077.0

Step 2

Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: diario, agregado: tipo de pago)



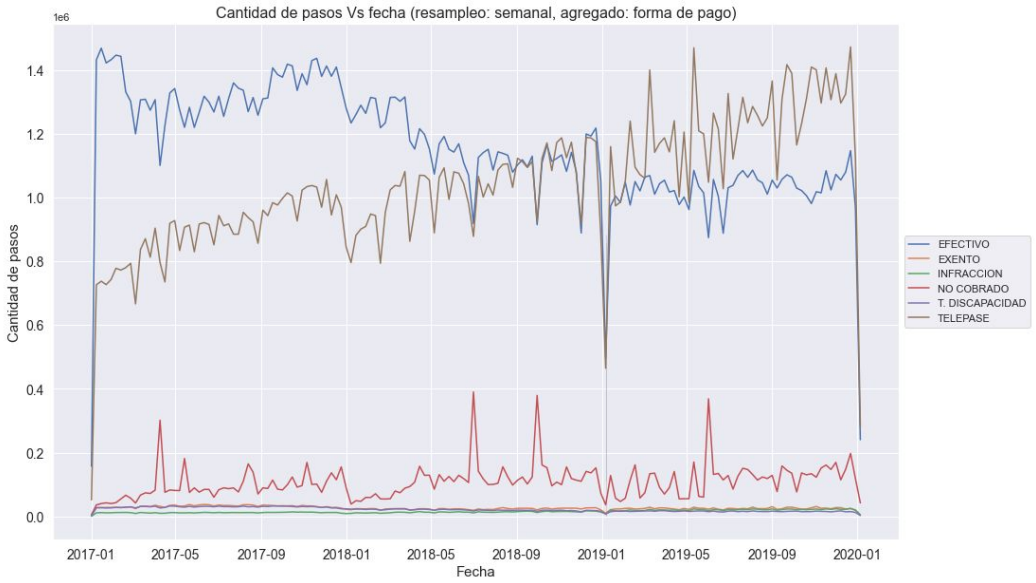
		cantidad_pasos							
		count	mean	std	min	25%	50%	75%	max
forma_pago	periodo								
EFECTIVO	2017	365.0	190419.008219	23131.664203	48.0	182460.00	193584.0	204371.00	234908.0
	2018	363.0	165905.231405	19699.905076	87321.0	155405.50	166053.0	178761.50	211039.0
	2019	364.0	147311.445055	15923.459468	88581.0	139588.25	149012.5	156786.00	184652.0
EXENTO	2017	365.0	4595.882192	1625.078710	1.0	2863.00	5243.0	5939.00	7458.0
	2018	363.0	3444.964187	1108.565454	1332.0	2173.50	4019.0	4257.00	6228.0
	2019	364.0	3600.879121	1142.542567	1396.0	2335.75	4161.5	4448.75	5527.0
INFRACCION	2017	364.0	1729.532967	500.653195	643.0	1223.00	1959.0	2089.00	2588.0
	2018	363.0	1984.286501	518.314364	732.0	1543.00	2081.0	2390.50	3231.0
	2019	364.0	3017.741758	652.394449	1324.0	2509.75	3206.0	3525.25	4124.0
NO COBRADO	2017	365.0	13275.808219	14172.860130	1785.0	5500.00	11118.0	17603.00	203456.0
	2018	365.0	16637.063014	20932.145077	1674.0	6710.00	14282.0	22587.00	273787.0
	2019	365.0	17171.734247	17890.178470	2585.0	9219.00	13348.0	22839.00	303470.0
T. DISCAPACIDAD	2017	364.0	4299.722527	661.242759	2147.0	3825.75	4493.5	4839.50	5289.0
	2018	363.0	2924.460055	474.627863	1240.0	2632.00	2951.0	3281.00	3806.0
	2019	364.0	2312.337912	341.827432	1248.0	2077.50	2379.5	2566.75	2943.0
TELEPASE	2017	365.0	127706.021918	36027.863123	49547.0	95091.00	144164.0	155668.00	176709.0
	2018	363.0	148106.462810	39415.708444	61379.0	110312.50	166501.0	179202.50	197787.0
	2019	364.0	174643.233516	46616.531859	69624.0	132454.00	188982.5	214376.00	246502.0

Step 2

Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: semanal, agregados: tipo de pago)



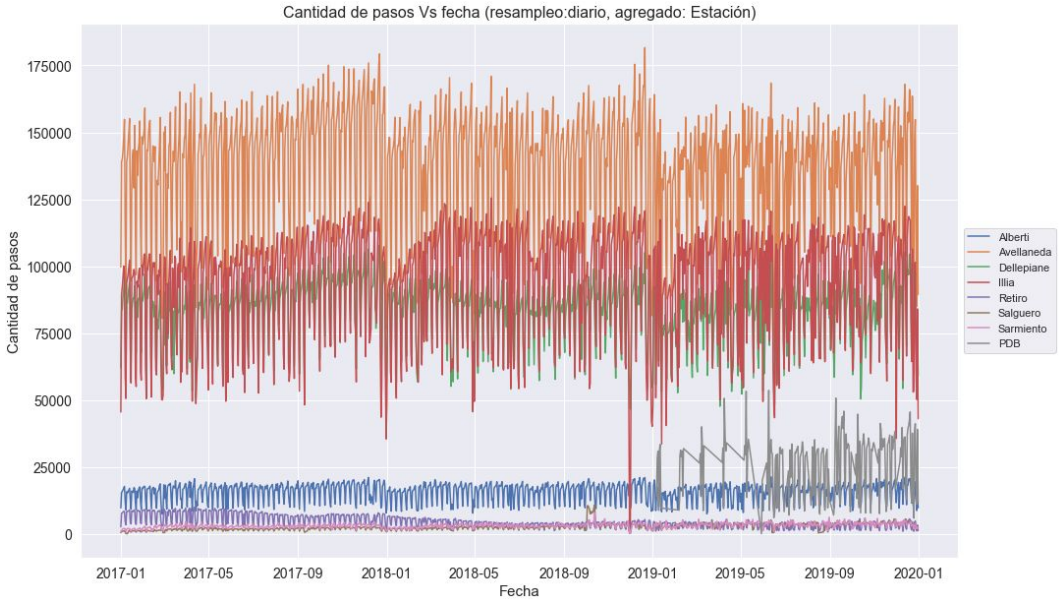
		cantidad_pasos							
		count	mean	std	min	25%	50%	75%	max
forma_pago	periodo								
EFECTIVO	2017	53.0	1.311376e+06	177839.177724	157472.0	1280462.0	1327848.0	1388160.0	1468278.0
	2018	53.0	1.136294e+06	172216.119882	109867.0	1108643.0	1143234.0	1215682.0	1315272.0
	2019	53.0	1.011724e+06	119298.379352	240871.0	1004718.0	1029880.0	1056881.0	1146883.0
EXENTO	2017	53.0	3.165089e+04	5461.474729	1638.0	28818.0	32925.0	34693.0	37705.0
	2018	53.0	2.359475e+04	3698.950116	1912.0	22508.0	24325.0	25353.0	27966.0
	2019	53.0	2.473057e+04	3832.595262	5286.0	23375.0	25183.0	26473.0	31474.0
INFRACCION	2017	53.0	1.187830e+04	1889.142455	670.0	11598.0	12211.0	12679.0	14358.0
	2018	53.0	1.359049e+04	2568.532959	1388.0	12535.0	13767.0	15213.0	17773.0
	2019	53.0	2.072562e+04	3051.235508	5035.0	19542.0	21270.0	22581.0	25235.0
NO COBRADO	2017	53.0	9.142774e+04	45110.287493	5756.0	72238.0	84815.0	100593.0	301947.0
	2018	53.0	1.145760e+05	63992.385984	5814.0	85489.0	109258.0	129262.0	390385.0
	2019	53.0	1.182582e+05	52160.945342	42468.0	76771.0	126381.0	141195.0	369017.0
T. DISCAPACIDAD	2017	53.0	2.953017e+04	4309.338151	2982.0	28342.0	30496.0	31465.0	33014.0
	2018	53.0	2.002979e+04	3328.980032	1598.0	19269.0	20040.0	22044.0	24023.0
	2019	53.0	1.588096e+04	2110.980359	3103.0	15335.0	16183.0	17003.0	18224.0
TELEPASE	2017	53.0	8.794849e+05	149791.301654	51924.0	829802.0	915602.0	960410.0	1056733.0
	2018	53.0	1.014390e+06	167096.779078	72776.0	948334.0	1054663.0	1103902.0	1188376.0
	2019	53.0	1.199437e+06	190685.503603	281529.0	1120587.0	1224162.0	1310210.0	1472070.0

Step 2

Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: diario, agregado: estación)



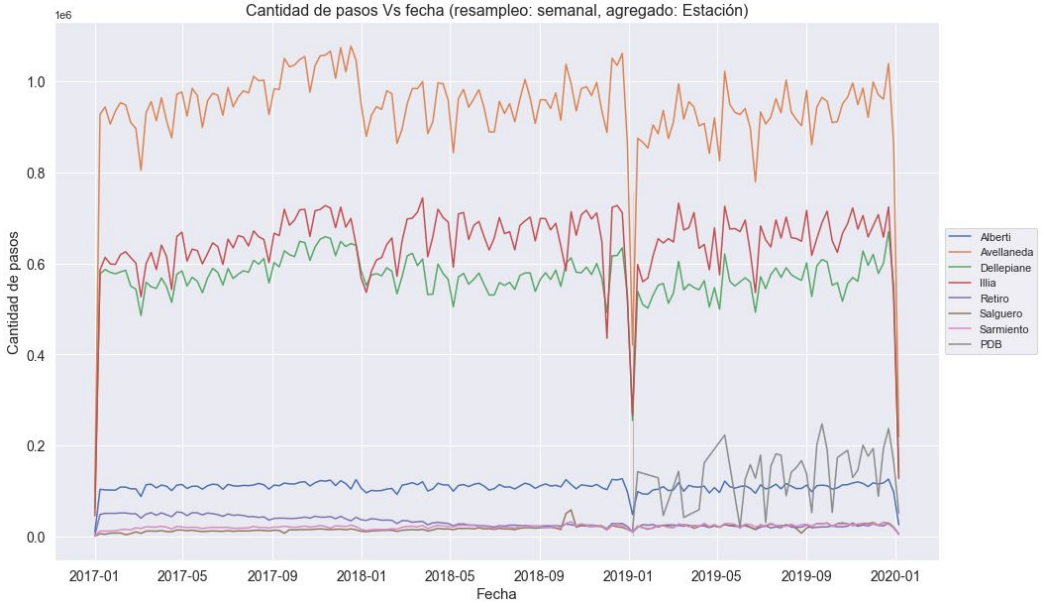
		cantidad_pasos							
		count	mean	std	min	25%	50%	75%	max
estacion	periodo								
Alberti	2017	365.0	15861.682192	3278.391490	7347.0	14209.00	17088.0	18212.00	21024.0
	2018	365.0	15711.895890	3375.428980	7685.0	13564.00	16906.0	18187.00	21020.0
	2019	365.0	15428.073973	3353.699960	7437.0	13590.00	16531.0	17882.00	21303.0
Avellaneda	2017	365.0	139209.835616	23143.078178	76421.0	124684.00	145869.0	155460.00	179343.0
	2018	365.0	136032.931507	23465.741979	66780.0	118680.00	144526.0	153334.00	181648.0
	2019	365.0	132338.504110	21922.418379	68460.0	118755.00	138739.0	148169.00	168417.0
Dellepiane	2017	365.0	84092.969863	11756.172051	50802.0	77438.00	86509.0	92473.00	107973.0
	2018	365.0	81537.673973	11293.792757	46629.0	75088.00	84752.0	89129.00	105118.0
	2019	365.0	80261.263014	11714.063813	47628.0	74284.00	82711.0	88359.00	105392.0
Illia	2017	365.0	91974.158904	20101.004502	35435.0	74283.00	99296.0	107263.00	123854.0
	2018	365.0	94491.668493	22427.688890	387.0	74194.00	103680.0	112327.00	125519.0
	2019	365.0	93575.778082	20395.165583	33529.0	75908.00	102459.0	109905.00	122390.0
PDB	2019	216.0	26242.481481	11265.602172	14.0	14501.75	30310.5	33486.50	53568.0
Retiro	2017	365.0	6446.104110	2164.232047	1491.0	4139.00	7120.0	8413.00	9895.0
	2018	364.0	3842.530220	1263.428290	260.0	2791.25	3960.0	4589.00	6649.0
	2019	365.0	3206.301370	1128.797346	892.0	2130.00	3610.0	4051.00	5585.0
Salguero	2017	355.0	1737.543662	532.346931	15.0	1361.00	1810.0	2123.00	2942.0
	2018	363.0	2622.741047	1362.665048	75.0	1954.00	2455.0	2941.50	10766.0
	2019	355.0	3440.507042	894.454160	350.0	2883.50	3519.0	4047.50	5819.0
Sarmiento	2017	363.0	2749.834711	676.528027	737.0	2360.00	2795.0	3136.50	4554.0
	2018	363.0	3038.187328	820.976307	113.0	2553.00	3046.0	3516.50	7740.0
	2019	364.0	3474.390110	871.820101	849.0	2957.25	3476.0	4032.25	6188.0

Step 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs fecha (resampleo: semanal, agregados: estación)

Etapa 2



		cantidad_pasos							
		count	mean	std	min	25%	50%	75%	max
estacion	periodo								
Alberti	2017	53.0	109236.113208	15562.868002	9638.0	105339.0	111453.0	114777.0	124861.0
	2018	53.0	108204.566038	15783.166272	9063.0	104294.0	110774.0	114081.0	126838.0
	2019	53.0	106249.943396	13884.245059	25629.0	102274.0	108703.0	113156.0	125806.0
Avellaneda	2017	53.0	958709.245283	132958.125439	99631.0	932110.0	972611.0	1011838.0	1078510.0
	2018	53.0	936830.566038	128479.476265	85601.0	915263.0	959324.0	984854.0	1062382.0
	2019	53.0	911387.811321	111411.007198	219405.0	902705.0	927571.0	956925.0	1039883.0
Dellepiane	2017	53.0	579130.830189	81785.034073	62026.0	561110.0	583798.0	615316.0	659108.0
	2018	53.0	561533.037736	76868.124452	54639.0	551763.0	573816.0	585726.0	634812.0
	2019	53.0	552742.660377	69207.315546	141108.0	541990.0	559436.0	578388.0	670699.0
Illia	2017	53.0	633406.943396	95037.593847	45477.0	605508.0	640626.0	671795.0	727544.0
	2018	53.0	650744.509434	103635.092846	40103.0	638604.0	673936.0	701209.0	744460.0
	2019	53.0	644436.962264	89481.737141	127053.0	633384.0	656928.0	687950.0	732906.0
PDB	2019	41.0	138253.073171	62058.041954	13992.0	89252.0	151548.0	181425.0	247650.0
Retiro	2017	53.0	44392.981132	7719.374527	2532.0	40484.0	44207.0	49864.0	53557.0
	2018	53.0	26390.207547	6024.661706	1205.0	23238.0	24831.0	29338.0	38875.0
	2019	53.0	22081.132075	3766.908782	4379.0	20314.0	22854.0	24019.0	29557.0
Salguero	2017	53.0	11638.264151	3507.755167	401.0	10617.0	12023.0	14149.0	16454.0
	2018	53.0	17963.301887	8258.568808	1312.0	14275.0	16454.0	19035.0	58395.0
	2019	53.0	23044.905660	5151.662298	5281.0	20121.0	23741.0	26691.0	30405.0
Sarmiento	2017	53.0	18833.773585	3981.452859	737.0	17865.0	19787.0	21135.0	24765.0
	2018	53.0	20808.716981	4650.840223	1432.0	18849.0	21522.0	23686.0	32441.0
	2019	53.0	23861.849057	4314.021564	4655.0	21735.0	24326.0	26928.0	29211.0

Step 2

EDA - multivariado (totalidad dataset)

Bonus: procedimiento para graficar stacked-bars

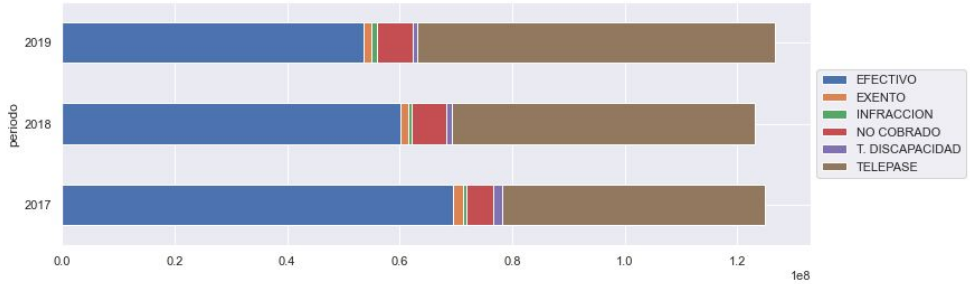
Etapa 2

	forma_pago	periodo	cantidad_pasos
17	TELEPASE	2019	63570137
2	EFECTIVO	2019	53621366
11	NO COBRADO	2019	6267683
5	EXENTO	2019	1310720
8	INFRACCION	2019	1098458
14	T. DISCAPACIDAD	2019	841691
1	EFECTIVO	2018	60223599
16	TELEPASE	2018	53762646
10	NO COBRADO	2018	6072528
4	EXENTO	2018	1250522
13	T. DISCAPACIDAD	2018	1061579
7	INFRACCION	2018	720296
0	EFECTIVO	2017	69502938
15	TELEPASE	2017	46612698
9	NO COBRADO	2017	4845670
3	EXENTO	2017	1677497
12	T. DISCAPACIDAD	2017	1565099
6	INFRACCION	2017	629550

```
dataset_analisis=dataset_analisis.pivot_table(index=["periodo"],
                    columns='forma_pago',
                    values='cantidad_pasos')
dataset_analisis
```

forma_pago	EFECTIVO	EXENTO	INFRACCION	NO COBRADO	T. DISCAPACIDAD	TELEPASE
periodo						
2017	69502938	1677497	629550	4845670	1565099	46612698
2018	60223599	1250522	720296	6072528	1061579	53762646
2019	53621366	1310720	1098458	6267683	841691	63570137

```
dataset_analisis.plot(kind='barh', stacked=True, figsize=(12,4), legend='False')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```

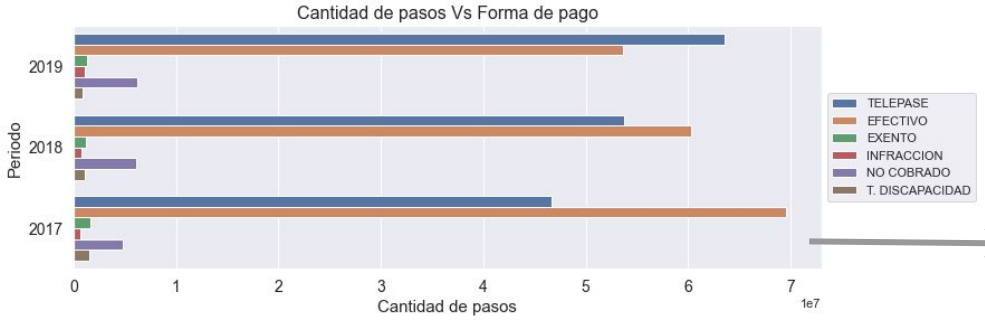


Step 2

EDA - multivariado (totalidad dataset)

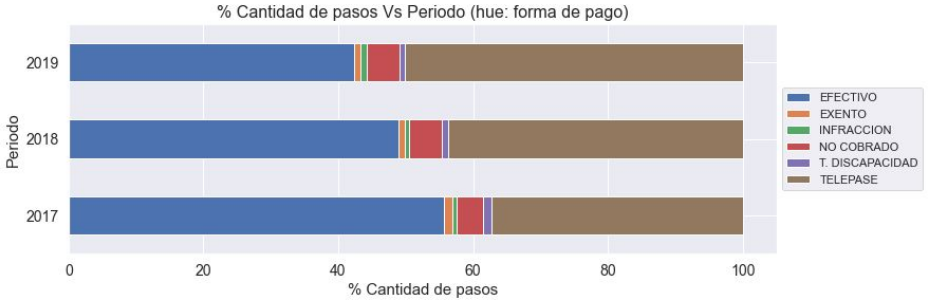
Cantidad de pasos Vs periodo (hue: forma de pago)

Etapa 2



forma_pago	periodo	cantidad_pasos
TELEPASE	2019	63570137
EFECTIVO	2019	53621366
NO COBRADO	2019	6267683
EXENTO	2019	1310720
INFRACCION	2019	1098458
T. DISCAPACIDAD	2019	841691
EFECTIVO	2018	60223599
TELEPASE	2018	53762646
NO COBRADO	2018	6072528
EXENTO	2018	1250522
T. DISCAPACIDAD	2018	1061579
INFRACCION	2018	720296

EFECTIVO	2017	69502938
TELEPASE	2017	46612698
NO COBRADO	2017	4845670
EXENTO	2017	1677497
T. DISCAPACIDAD	2017	1565099
INFRACCION	2017	629550



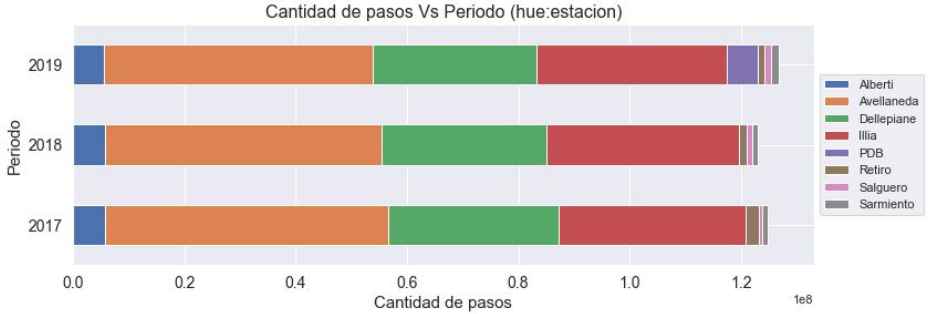
forma_pago	EFECTIVO	EXENTO	INFRACCION	NO COBRADO	T. DISCAPACIDAD	TELEPASE
periodo						
2017	55.7	1.3	0.5	3.9	1.3	37.3
2018	48.9	1.0	0.6	4.9	0.9	43.7
2019	42.3	1.0	0.9	4.9	0.7	50.2

Step 2

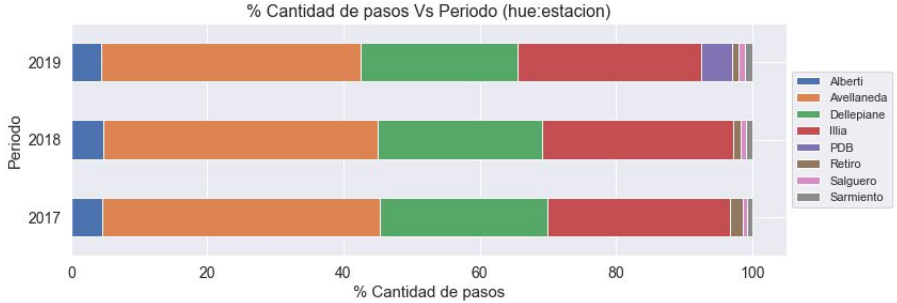
Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs periodo (hue: estacion)



estacion	Alberti	Avellaneda	Dellepiane	Illia	PDB	Retiro	Salguero	Sarmiento
periodo								
2017	5789514.0	50811590.0	30693934.0	33570568.0	NaN	2352828.0	616828.0	998190.0
2018	5734842.0	49652020.0	29761251.0	34489459.0	NaN	1398681.0	952055.0	1102862.0
2019	5631247.0	48303554.0	29295361.0	34155159.0	5668376.0	1170300.0	1221380.0	1264678.0



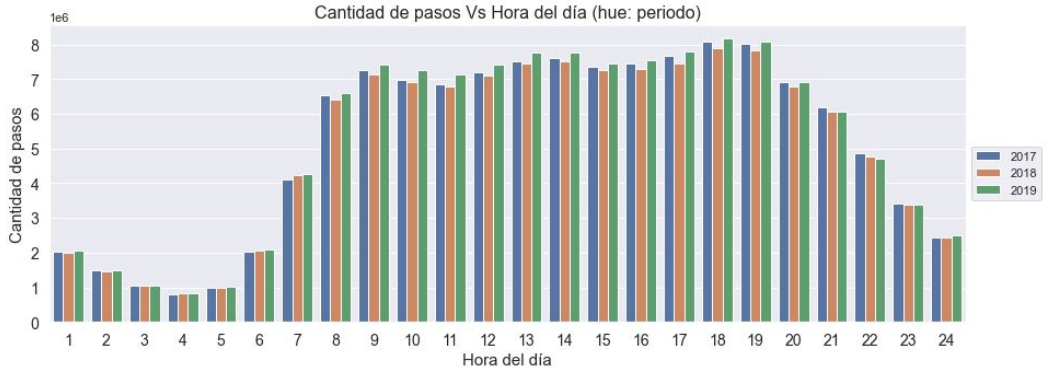
estacion	Alberti	Avellaneda	Dellepiane	Illia	PDB	Retiro	Salguero	Sarmiento
periodo								
2017	4.6	40.7	24.6	26.9	NaN	1.9	0.5	0.8
2018	4.7	40.3	24.2	28.0	NaN	1.1	0.8	0.9
2019	4.4	38.1	23.1	27.0	4.5	0.9	1.0	1.0

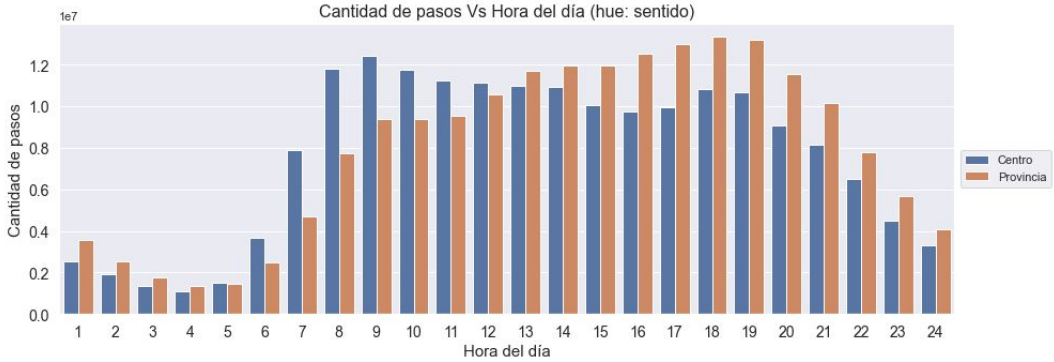
Step 2

Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs hora del día (hue: periodo - sentido)



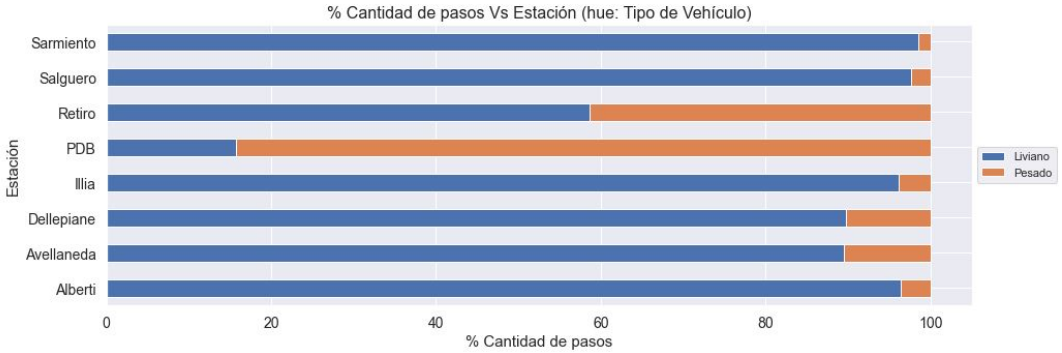


Step 2

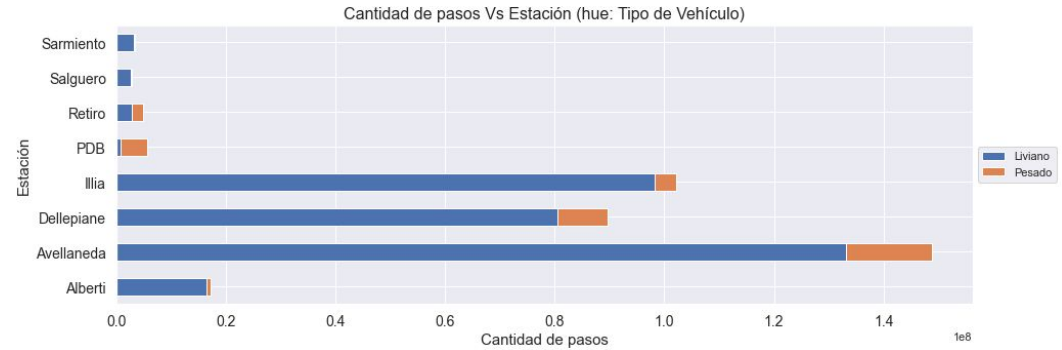
Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs estación (hue: tipo vehiculo) (periodo 2017 - 2019)



tipo_vehiculo	Liviano	Pesado
estacion		
Alberti	96.4	3.6
Avellaneda	89.5	10.5
Dellepiane	89.7	10.3
Illia	96.1	3.9
PDB	15.7	84.3
Retiro	58.7	41.3
Salguero	97.6	2.4
Sarmiento	98.5	1.5



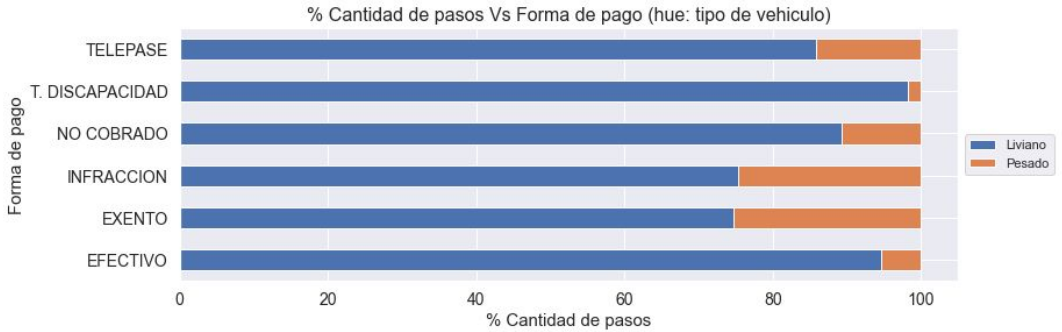
tipo_vehiculo	Liviano	Pesado
estacion		
Alberti	16542845	612758
Avellaneda	133116334	15650830
Dellepiane	80467381	9283165
Illia	98218763	3996423
PDB	890931	4777445
Retiro	2886954	2034855
Salguero	2723682	66581
Sarmiento	3315343	50387

Step 2

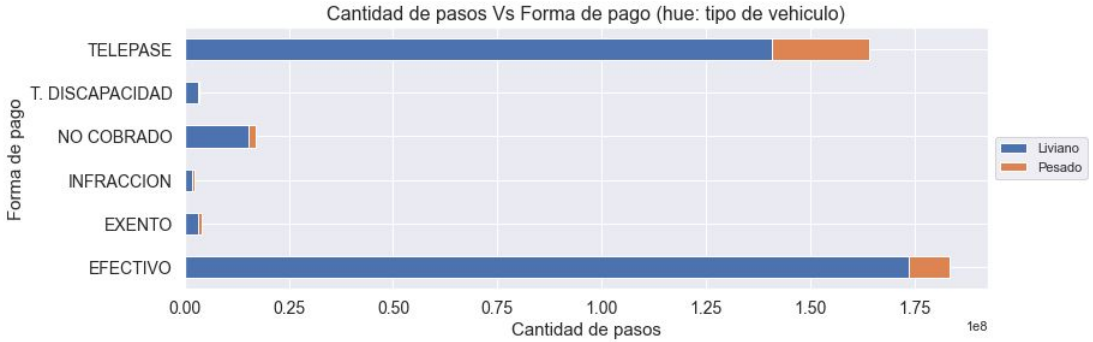
Etapa 2

EDA - multivariado (totalidad dataset)

Cantidad de pasos Vs forma de pago (hue: tipo de vehículo)



forma_pago	tipo_vehiculo Liviano	Pesado
EFECTIVO	94.7	5.3
EXENTO	74.7	25.3
INFRACCION	75.3	24.7
NO COBRADO	89.3	10.7
T. DISCAPACIDAD	98.3	1.7
TELEPASE	85.9	14.1



forma_pago	tipo_vehiculo Liviano	Pesado
EFECTIVO	173541605	9806298
EXENTO	3166414	1072325
INFRACCION	1844773	603531
NO COBRADO	15346681	1839200
T. DISCAPACIDAD	3408349	60020
TELEPASE	140854411	23091070

Step 3

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Dataset base (etapa 1)

Etapa 2

```
dataset
```

	fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num	estacion	sentido	tipo_vehiculo	forma_pago	cantidad_pasos
0	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	25
1	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Alberti	Centro	Liviano	TELEPASE	7
2	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	NO COBRADO	5
3	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	2
4	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Alberti	Centro	Liviano	EFECTIVO	94
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
812148	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	NO COBRADO	7
812149	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Liviano	TELEPASE	4
812150	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Salguero	Provincia	Pesado	NO COBRADO	1
812151	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	NO COBRADO	7
812152	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Sarmiento	Provincia	Liviano	TELEPASE	7

3560648 rows × 14 columns

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Dataset filtrado (Est: Illia, T. pago: efectivo)

```
dataset_Illia_efectivo =  dataset[(dataset.estacion=='Illia') & (dataset.forma_pago=='EFECTIVO')]\
                .drop(columns=['estacion', 'forma_pago']).reset_index(drop=True)
dataset_Illia_efectivo
```

	fecha_comp	fecha	fecha_num	periodo	mes	hora_inicio	hora_fin	dia	dia_num	sentido	tipo_vehiculo	cantidad_pasos
0	2017-01-01 01:00:00	2017-01-01	736330	2017	1	0	1	Domingo	6	Centro	Liviano	1
1	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Centro	Liviano	10
2	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Centro	Liviano	783
3	2017-01-01 02:00:00	2017-01-01	736330	2017	1	1	2	Domingo	6	Centro	Pesado	3
4	2017-01-01 03:00:00	2017-01-01	736330	2017	1	2	3	Domingo	6	Centro	Liviano	14
...	...	...	...	...	...	...	...	...	...	...	...	...
238857	2019-12-31 22:00:00	2019-12-31	737424	2019	12	21	22	Martes	1	Provincia	Liviano	556
238858	2019-12-31 22:00:00	2019-12-31	737424	2019	12	21	22	Martes	1	Provincia	Pesado	1
238859	2019-12-31 23:00:00	2019-12-31	737424	2019	12	22	23	Martes	1	Centro	Liviano	214
238860	2019-12-31 23:00:00	2019-12-31	737424	2019	12	22	23	Martes	1	Provincia	Liviano	190
238861	2020-01-01 00:00:00	2019-12-31	737424	2019	12	23	24	Martes	1	Centro	Liviano	4

238862 rows × 12 columns

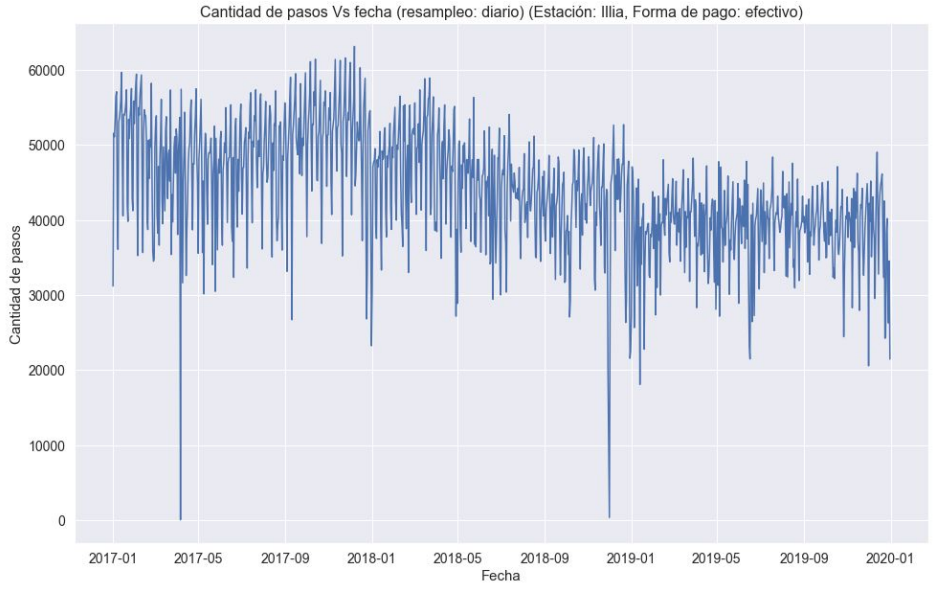
Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Cantidad de pasos Vs periodo (resampleo diario, no agregado)

```
dataset_Illia_efectivo_diario =  dataset_Illia_efectivo[['fecha','fecha_num','periodo','mes', 'dia','cantidad_pasos']]\
        .groupby([pd.Grouper(key= 'fecha', freq='D'),'fecha_num','periodo','mes', 'dia' ])\
        .sum().reset_index()
dataset_Illia_efectivo_diario
```



	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1086	2019-12-27	737420	2019	12	Viernes	40182
1087	2019-12-28	737421	2019	12	Sábado	30681
1088	2019-12-29	737422	2019	12	Domingo	26259
1089	2019-12-30	737423	2019	12	Lunes	34523
1090	2019-12-31	737424	2019	12	Martes	21447

1091 rows × 6 columns

Step 3

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Valores nulos y valores cero

Etapa 2

dataset_Illia_efectivo_diario

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1086	2019-12-27	737420	2019	12	Viernes	40182
1087	2019-12-28	737421	2019	12	Sábado	30681
1088	2019-12-29	737422	2019	12	Domingo	26259
1089	2019-12-30	737423	2019	12	Lunes	34523
1090	2019-12-31	737424	2019	12	Martes	21447

1091 rows × 6 columns

```
dataset_Illia_efectivo_diario.isnull().sum()
```

```
fecha            0
fecha_num        0
periodo          0
mes              0
dia              0
cantidad_pasos   0
dtype: int64
```

```
(dataset_Illia_efectivo_diario.cantidad_pasos==0).sum()
```

```
0
```

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Valores faltantes

```
df_validacion = pd.date_range(start='2017-01-01', end='2019-12-31',freq='D').to_frame().rename(columns={0:'fecha'})
df_validacion
```

	fecha
2017-01-01	2017-01-01
2017-01-02	2017-01-02
2017-01-03	2017-01-03
2017-01-04	2017-01-04
2017-01-05	2017-01-05
...	...
2019-12-27	2019-12-27
2019-12-28	2019-12-28
2019-12-29	2019-12-29
2019-12-30	2019-12-30
2019-12-31	2019-12-31

1095 rows × 1 columns

```
dataset_Illia_efectivo_diario
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1086	2019-12-27	737420	2019	12	Viernes	40182
1087	2019-12-28	737421	2019	12	Sábado	30681
1088	2019-12-29	737422	2019	12	Domingo	26259
1089	2019-12-30	737423	2019	12	Lunes	34523
1090	2019-12-31	737424	2019	12	Martes	21447

1091 rows × 6 columns

```
merge_validation = df_validacion.merge(dataset_Illia_efectivo_diario, on='fecha', how='left')
merge_validation
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330.0	2017.0	1.0	Domingo	31183.0
1	2017-01-02	736331.0	2017.0	1.0	Lunes	51568.0
2	2017-01-03	736332.0	2017.0	1.0	Martes	51093.0
3	2017-01-04	736333.0	2017.0	1.0	Miércoles	53332.0
4	2017-01-05	736334.0	2017.0	1.0	Jueves	56486.0
...	...	...	...	...	...	...
1090	2019-12-27	737420.0	2019.0	12.0	Viernes	40182.0
1091	2019-12-28	737421.0	2019.0	12.0	Sábado	30681.0
1092	2019-12-29	737422.0	2019.0	12.0	Domingo	26259.0
1093	2019-12-30	737423.0	2019.0	12.0	Lunes	34523.0
1094	2019-12-31	737424.0	2019.0	12.0	Martes	21447.0

1095 rows × 6 columns

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Valores faltantes

```
merge_validation = df_validacion.merge(dataset_Illia_efectivo_diario, on='fecha', how='left')
merge_validation
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330.0	2017.0	1.0	Domingo	31183.0
1	2017-01-02	736331.0	2017.0	1.0	Lunes	51568.0
2	2017-01-03	736332.0	2017.0	1.0	Martes	51093.0
3	2017-01-04	736333.0	2017.0	1.0	Miércoles	53332.0
4	2017-01-05	736334.0	2017.0	1.0	Jueves	56486.0
...	...	...	...	...	...	...
1090	2019-12-27	737420.0	2019.0	12.0	Viernes	40182.0
1091	2019-12-28	737421.0	2019.0	12.0	Sábado	30681.0
1092	2019-12-29	737422.0	2019.0	12.0	Domingo	26259.0
1093	2019-12-30	737423.0	2019.0	12.0	Lunes	34523.0
1094	2019-12-31	737424.0	2019.0	12.0	Martes	21447.0

1095 rows × 6 columns

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
540	2018-06-25	NaN	NaN	NaN	NaN	NaN
632	2018-09-25	NaN	NaN	NaN	NaN	NaN
698	2018-11-30	NaN	NaN	NaN	NaN	NaN
878	2019-05-29	NaN	NaN	NaN	NaN	NaN

```
merge_validation.cantidad_pasos.iloc[indices_a_reemplazar]=\
merge_validation.cantidad_pasos.iloc[indices_a_reemplazar].replace(np.nan, 0)
```

```
merge_validation.fecha_num.iloc[indices_a_reemplazar]=\
merge_validation.fecha.iloc[indices_a_reemplazar].map(datetime.datetime.toordinal)
```

```
merge_validation.periodo.iloc[indices_a_reemplazar]=\
merge_validation.fecha.iloc[indices_a_reemplazar].dt.year
```

```
merge_validation.mes.iloc[indices_a_reemplazar]=\
merge_validation.fecha.iloc[indices_a_reemplazar].dt.month
```

```
merge_validation.dia.iloc[indices_a_reemplazar]=\
merge_validation.fecha.iloc[indices_a_reemplazar].dt.weekday
```

```
merge_validation.dia.iloc[indices_a_reemplazar] = merge_validation.dia.iloc[indices_a_reemplazar]\
        .replace({0: 'Lunes', 1: 'Martes', 2: 'Miércoles', 3: 'Jueves', 4: 'Viernes', 5: 'Sábado', 6: 'Domingo'})
```

```
merge_validation.iloc[indices_a_reemplazar]
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
540	2018-06-25	736870.0	2018.0	6.0	Lunes	0.0
632	2018-09-25	736962.0	2018.0	9.0	Martes	0.0
698	2018-11-30	737028.0	2018.0	11.0	Viernes	0.0
878	2019-05-29	737208.0	2019.0	5.0	Miércoles	0.0

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Valores faltantes

```
merge_validation = merge_validation.astype({'cantidad_pasos': int, 'fecha_num': int, 'periodo': int, 'mes': int})
merge_validation
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns

```
dataset_Illia_efectivo_diario_no_faltantes = merge_validation
dataset_Illia_efectivo_diario_no_faltantes
```

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Valores faltantes

dataset_Illia_efectivo_diario

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1086	2019-12-27	737420	2019	12	Viernes	40182
1087	2019-12-28	737421	2019	12	Sábado	30681
1088	2019-12-29	737422	2019	12	Domingo	26259
1089	2019-12-30	737423	2019	12	Lunes	34523
1090	2019-12-31	737424	2019	12	Martes	21447

1091 rows × 6 columns

dataset_Illia_efectivo_diario_no_faltantes

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns

Step 3

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

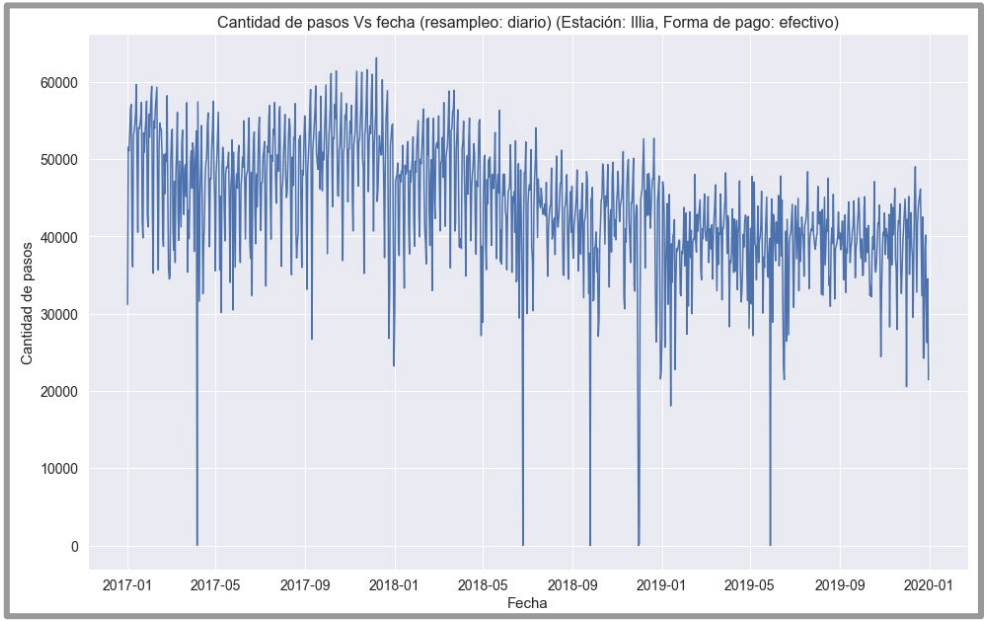
Outliers

Etapa 2

dataset_Illia_efectivo_diario_no_faltantes

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns



Step 3

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Outliers

Etapa 2

```
plotMovingAverage(dataset_analisis, 7, plot_intervals=True, plot_anomalies=True)
```


```python
def plotMovingAverage(series, window, plot_intervals=False, scale=1.96,
                      plot_anomalies=False):

    """
        series - dataframe with timeseries
        window - rolling window size
        plot_intervals - show confidence intervals
        plot_anomalies - show anomalies

    """
    global rolling_mean
    rolling_mean = series.rolling(window=window).mean()

    plt.figure(figsize=(15,10))
    plt.plot(series[window:], label="Actual values")
    plt.title("Moving average\n window size = {}".format(window))
    plt.plot(rolling_mean, "yellow", label="Rolling mean trend")

    # Plot confidence intervals for smoothed values
    if plot_intervals:
        mae = mean_absolute_error(series[window:], rolling_mean[window:])
        deviation = np.std(series[window:] - rolling_mean[window:])
        lower_bond = rolling_mean - (mae + scale * 1.5*deviation)
        upper_bond = rolling_mean + (mae + scale * 1.5* deviation)
        plt.plot(upper_bond, "r--", label="Upper Bond / Lower Bond")
        plt.plot(lower_bond, "r--")

        # Having the intervals, find abnormal values
        if plot_anomalies:
            global anomalies
            anomalies = pd.DataFrame(index=series.index, columns=series.columns)
            anomalies[series<lower_bond] = series[series<lower_bond]
            anomalies[series>upper_bond] = series[series>upper_bond]
            plt.plot(anomalies, "ro", markersize=10)

    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.grid(True)
```

Step 3

Etapa 2

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Outliers

```
plotMovingAverage(dataset_analisis, 7, plot_intervals=True, plot_anomalies=True)
```


```
anomalies.reset_index().dropna()
```

	fecha	cantidad_pasos
95	2017-04-06	40
540	2018-06-25	0
632	2018-09-25	0
698	2018-11-30	0
699	2018-12-01	350
878	2019-05-29	0

```
rolling_mean.reset_index()
```

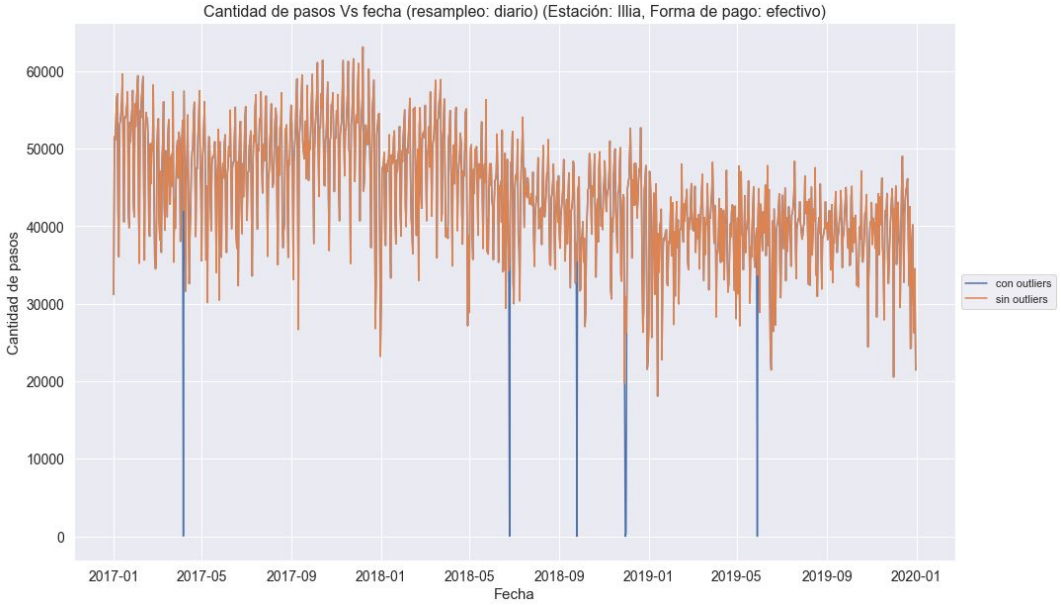
```
95     42084
540    34945
632    35539
698    30923
699    26167
878    33679
Name: cantidad_pasos, dtyp
```

Step 3

Feature engineering (Est: Illia, T. pago: efectivo, res: diario)

Outliers

Etapa 2



dataset_Illia_efectivo_diario_no_faltantes_no_outliers

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns

Step 4

Etapa 2

Dataset base para modelos de predicción

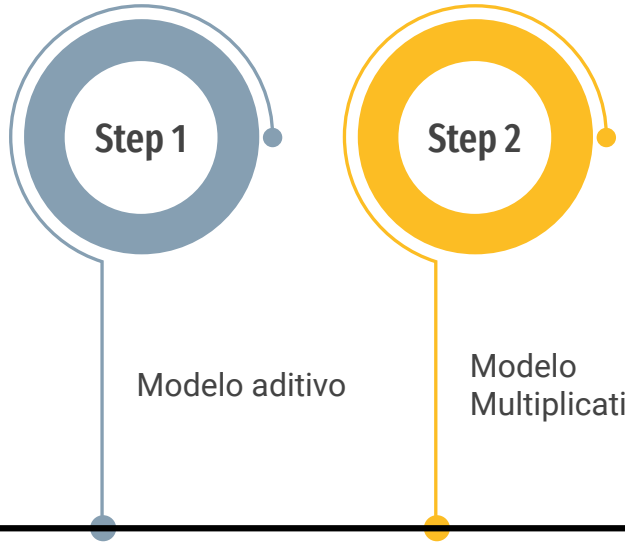
`dataset_Illia_efectivo_diario_no_faltantes_no_outliers`

	fecha	fecha_num	periodo	mes	dia	cantidad_pasos
0	2017-01-01	736330	2017	1	Domingo	31183
1	2017-01-02	736331	2017	1	Lunes	51568
2	2017-01-03	736332	2017	1	Martes	51093
3	2017-01-04	736333	2017	1	Miércoles	53332
4	2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...	...
1090	2019-12-27	737420	2019	12	Viernes	40182
1091	2019-12-28	737421	2019	12	Sábado	30681
1092	2019-12-29	737422	2019	12	Domingo	26259
1093	2019-12-30	737423	2019	12	Lunes	34523
1094	2019-12-31	737424	2019	12	Martes	21447

1095 rows × 6 columns

Etapa 3

Componentes de la serie temporal

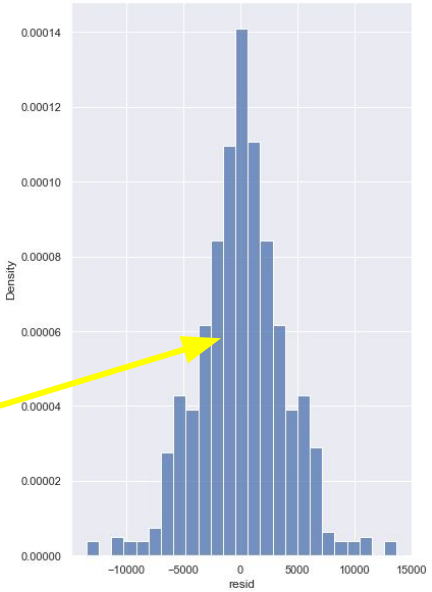


Step 1

Etapa 3

Componentes de la serie temporal

Modelo aditivo (Est: Illia, T. pago: efectivo, resampleo: diario)

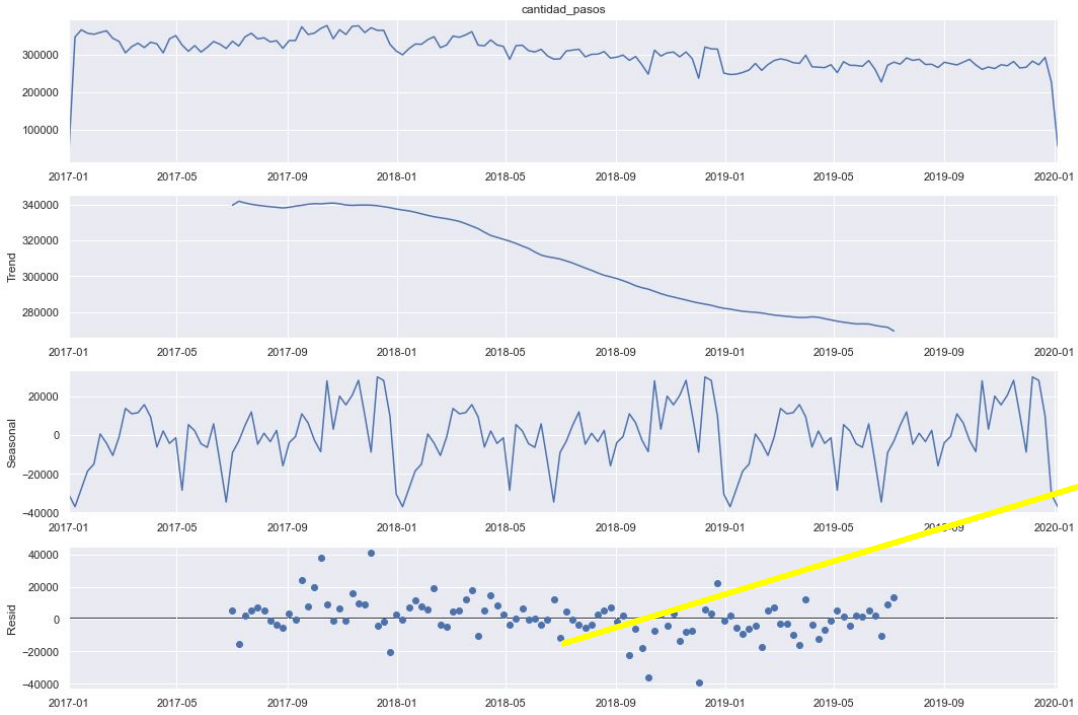


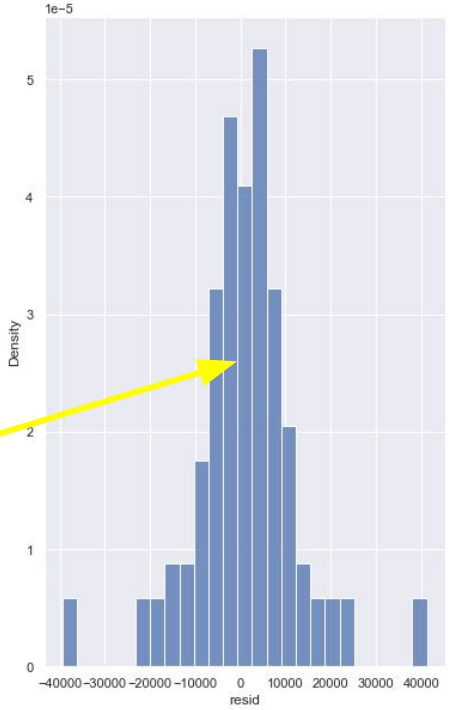
Step 1

Etapa 3

Componentes de la serie temporal

Modelo aditivo (Est: Illia, T. pago: efectivo, resampleo: semanal)



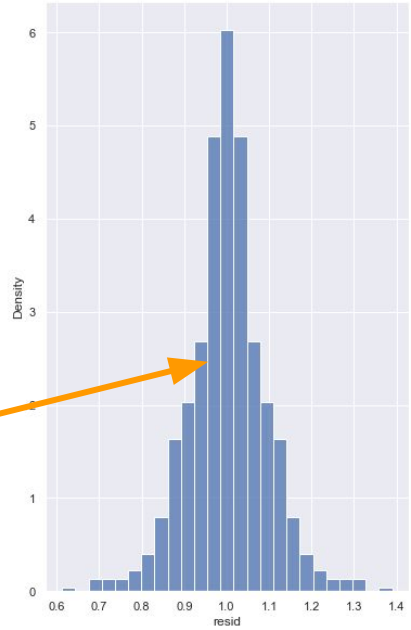


Step 2

Etapa 3

Componentes de la serie temporal

Modelo multiplicativo (Est: Illia, T. pago: efectivo, resampleo: diario)

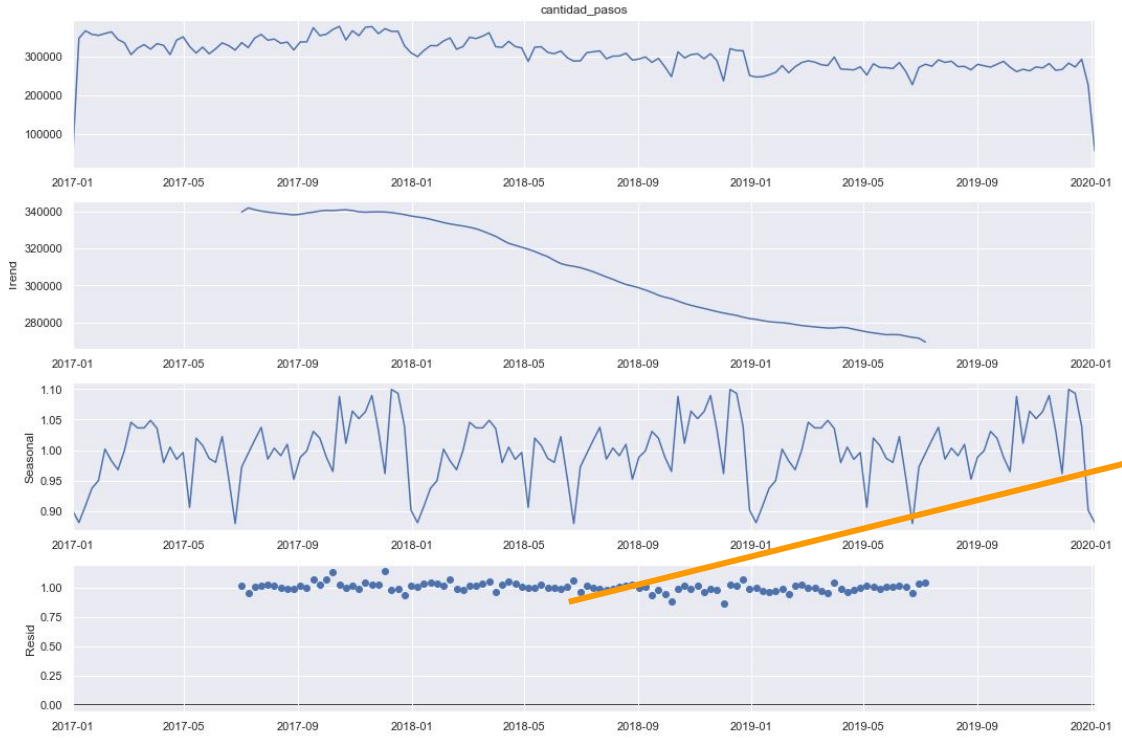


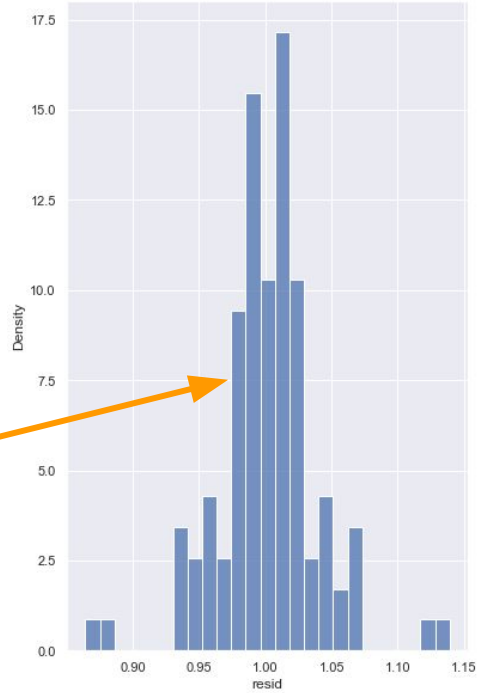
Step 2

Etapa 3

Componentes de la serie temporal

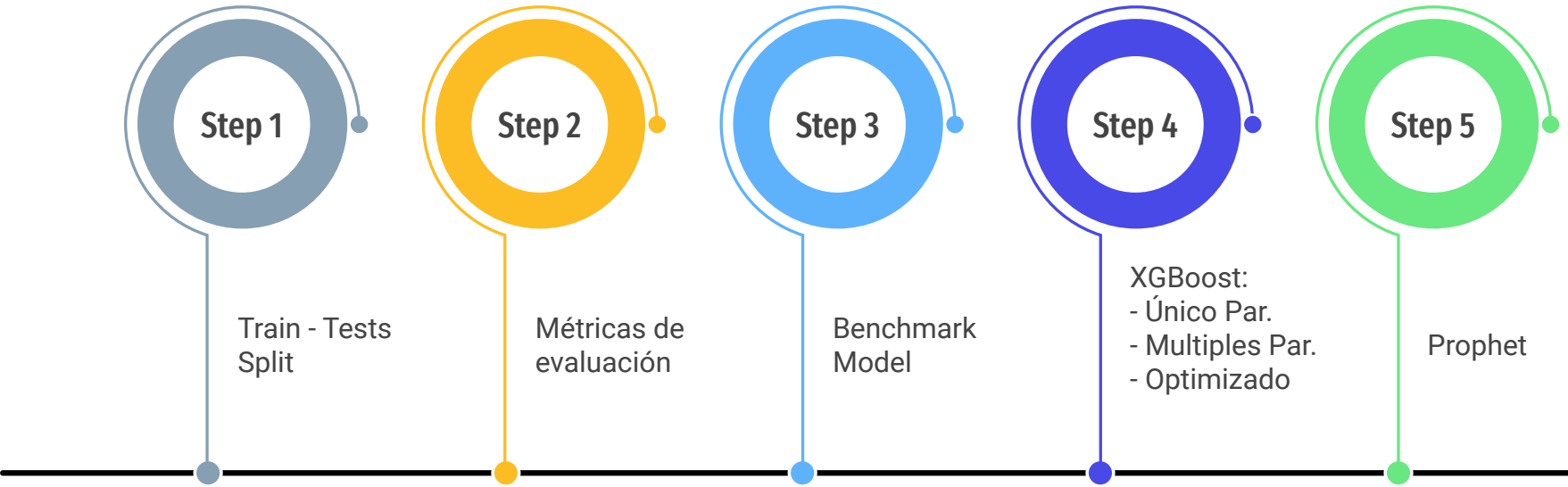
Modelo multiplicativo (Est: Illia, T. pago: efectivo, resampleo: semanal)





Etapa 4

Modelos de predicción



Step 1

Train - test split

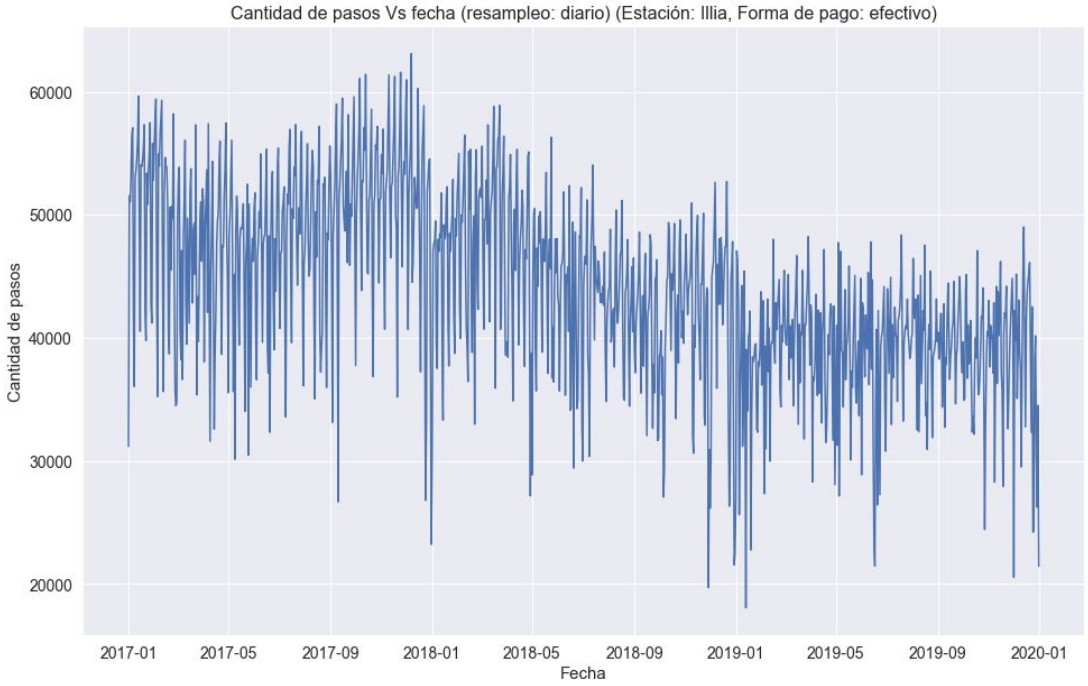
Único atributo (Est: Illia, T. pago: efectivo, res: diario)

Etapa 4

```
dataset_analisis = dataset_Illia_efectivo_diario_no_faltantes_no_outliers.\
                   set_index('fecha')[['fecha_num', 'cantidad_pasos']]
dataset_analisis
```

fecha	fecha_num	cantidad_pasos
2017-01-01	736330	31183
2017-01-02	736331	51568
2017-01-03	736332	51093
2017-01-04	736333	53332
2017-01-05	736334	56486
...	...	...
2019-12-27	737420	40182
2019-12-28	737421	30681
2019-12-29	737422	26259
2019-12-30	737423	34523
2019-12-31	737424	21447

1095 rows × 2 columns



Step 1

Train - test split

Único atributo (Est: Illia, T. pago: efectivo, res: diario)

Etapa 4

```
dataset_analisis = dataset_Illia_efectivo_diario_no_faltantes_no_outliers.\
                   set_index('fecha')[['fecha_num', 'cantidad_pasos']]
dataset_analisis
```

fecha	fecha_num	cantidad_pasos
2017-01-01	736330	31183
2017-01-02	736331	51568
2017-01-03	736332	51093
2017-01-04	736333	53332
2017-01-05	736334	56486
...	...	...
2019-12-27	737420	40182
2019-12-28	737421	30681
2019-12-29	737422	26259
2019-12-30	737423	34523
2019-12-31	737424	21447

1095 rows × 2 columns

```
df_train = dataset_analisis.loc['2017-01-01':'2019-09-30']
df_train
```

fecha	fecha_num	cantidad_pasos
2017-01-01	736330	31183
2017-01-02	736331	51568
2017-01-03	736332	51093
2017-01-04	736333	53332
2017-01-05	736334	56486
...	...	...
2019-09-26	737328	43110
2019-09-27	737329	44985
2019-09-28	737330	41925
2019-09-29	737331	38348
2019-09-30	737332	37170

1003 rows × 2 columns

```
df_test = dataset_analisis.loc['2019-10-01':]
df_test
```

fecha	fecha_num	cantidad_pasos
2019-10-01	737333	39669
2019-10-02	737334	34943
2019-10-03	737335	35054
2019-10-04	737336	43775
2019-10-05	737337	45150
...	...	...
2019-12-27	737420	40182
2019-12-28	737421	30681
2019-12-29	737422	26259
2019-12-30	737423	34523
2019-12-31	737424	21447

92 rows × 2 columns

```
X_train = df_train.fecha_num.values.reshape(-1, 1)
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.fecha_num.values.reshape(-1, 1)
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

Step 1

Train - test split

Múltiples atributos (Est: Illia, T. pago: efectivo, res: diario)

Etapa 4

```
dataset_analisis = dataset_Illia_efectivo_diario_no_faltantes_no_outliers.\
                   set_index('fecha')
dataset_analisis
```

fecha	fecha_num	periodo	mes	dia	cantidad_pasos
2017-01-01	736330	2017	1	Domingo	31183
2017-01-02	736331	2017	1	Lunes	51568
2017-01-03	736332	2017	1	Martes	51093
2017-01-04	736333	2017	1	Miércoles	53332
2017-01-05	736334	2017	1	Jueves	56486
...	...	...	...	...	...
2019-12-27	737420	2019	12	Viernes	40182
2019-12-28	737421	2019	12	Sábado	30681
2019-12-29	737422	2019	12	Domingo	26259
2019-12-30	737423	2019	12	Lunes	34523
2019-12-31	737424	2019	12	Martes	21447

1095 rows × 5 columns

```
#dataset_analisis= pd.get_dummies(dataset_analisis, columns=['periodo','mes','dia'], sparse=True)
dataset_analisis= pd.get_dummies(dataset_analisis, columns=['periodo','mes','dia'], sparse=True)
dataset_analisis
```

fecha	fecha_num	cantidad_pasos	periodo_2017	periodo_2018	periodo_2019	mes_1	mes_2	mes_3	mes_4	mes_5	...	mes_10	mes_11	mes_12	dia_Domingo	dia_Jueves	dia_Lunes	dia_Martes	dia_Miércoles	dia_Sábado	dia_Viernes
2017-01-01	736330	31183	1	0	0	1	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
2017-01-02	736331	51568	1	0	0	1	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
2017-01-03	736332	51093	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
2017-01-04	736333	53332	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
2017-01-05	736334	56486	1	0	0	1	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-12-27	737420	40182	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	1
2019-12-28	737421	30681	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	0	0	1	0
2019-12-29	737422	26259	0	0	1	0	0	0	0	0	...	0	0	1	1	0	0	0	0	0	0
2019-12-30	737423	34523	0	0	1	0	0	0	0	0	...	0	0	1	0	0	1	0	0	0	0
2019-12-31	737424	21447	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	1	0	0	0

1095 rows × 24 columns

Step 1

Etapa 4

Train - test split

Todos los atributos (Est: Illia, T. pago: efectivo, res: diario)

```
df_train = dataset_analisis.loc['2017-01-01':'2019-09-30']
df_train
```

```
df_test = dataset_analisis.loc['2019-10-01':]
df_test
```

```
X_train = df_train.drop(columns='cantidad_pasos')
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.drop(columns='cantidad_pasos')
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
#dataset_analisis= pd.get_dummies(dataset_analisis, columns=['periodo','mes','dia'], sparse=True)
dataset_analisis= pd.get_dummies(dataset_analisis, columns=['periodo','mes','dia'], sparse=True)
dataset_analisis
```

fecha	fecha_num	cantidad_pasos	periodo_2017	periodo_2018	periodo_2019	mes_1	mes_2	mes_3	mes_4	mes_5	...	mes_10	mes_11	mes_12	dia_Domingo	dia_Jueves	dia_Lunes	dia_Martes	dia_Miércoles	dia_Sábado	dia_Viernes
2017-01-01	736330	31183	1	0	0	1	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
2017-01-02	736331	51568	1	0	0	1	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
2017-01-03	736332	51093	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
2017-01-04	736333	53332	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
2017-01-05	736334	56486	1	0	0	1	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2019-12-27	737420	40182	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	1
2019-12-28	737421	30681	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	0	0	1	0
2019-12-29	737422	26259	0	0	1	0	0	0	0	0	...	0	0	1	1	0	0	0	0	0	0
2019-12-30	737423	34523	0	0	1	0	0	0	0	0	...	0	0	1	0	0	1	0	0	0	0
2019-12-31	737424	21447	0	0	1	0	0	0	0	0	...	0	0	1	0	0	0	1	0	0	0

1095 rows × 24 columns

Step 2

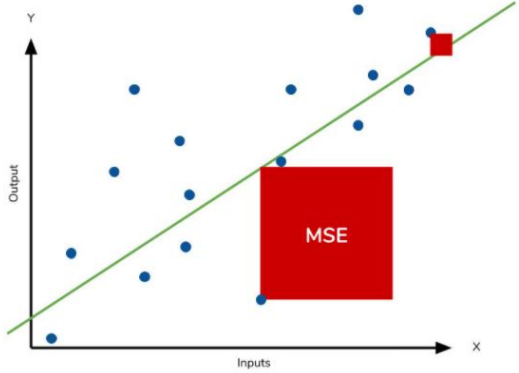
Etapa 4

Métricas de evaluación (útiles en TS)

Mean square error (MSE) and Root mean square error (RMSE)

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)}^{2}$$

The square of the difference between actual and predicted



$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values

$y_1, y_2, \ldots, y_n$ are observed values

n is the number of observations

Step 2

Etapa 4

Métricas de evaluación (útiles en TS)

Mean absolute error (MAE) and Mean absolute percentage error (MAPE)

$$MAE = \frac{1}{n}\sum\left|y - \hat{y}\right|$$

Divide by the total number of data points

Predicted output value

Actual output value

Sum of

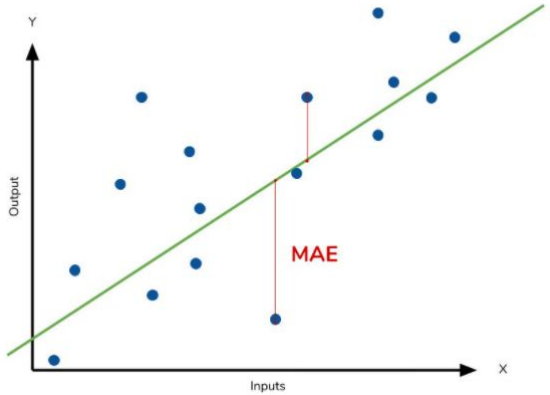
The absolute value of the residual

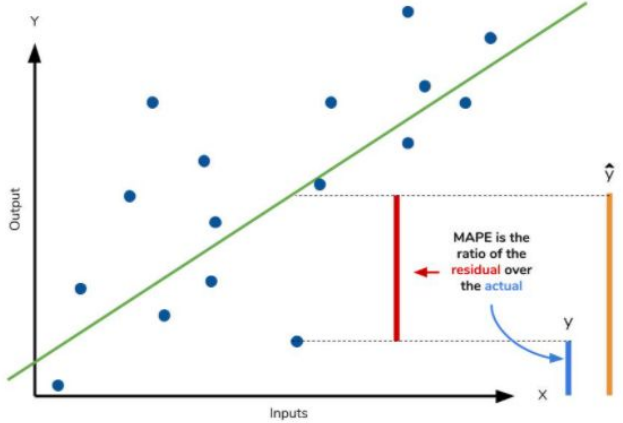
$$MAPE = \frac{100\%}{n}\sum\left|\frac{y - \hat{y}}{y}\right|$$

Multiplying by 100% converts to percentage

The residual

Each residual is scaled against the actual value





Step 2

Etapa 4

Métricas de evaluación (útiles en TS)

Scikit Learn regression metrics

squared : *bool, default=True*
If True returns MSE value, if False returns RMSE value.

```
rmse_train = round(mean_squared_error(y_train, y_train_pred, squared=False),1)
rmse_test = round(mean_squared_error(y_test, y_test_pred, squared=False),1)
```

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Regression metrics

See the Regression metrics section of the user guide for further details.

`metrics.explained_variance_score`(y_true, ...)	Explained variance regression score function.
`metrics.max_error`(y_true, y_pred)	max_error metric calculates the maximum residual error.
`metrics.mean_absolute_error`(y_true, y_pred, *)	Mean absolute error regression loss.
`metrics.mean_squared_error`(y_true, y_pred, *)	Mean squared error regression loss.
`metrics.mean_squared_log_error`(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
`metrics.median_absolute_error`(y_true, y_pred, *)	Median absolute error regression loss.
`metrics.mean_absolute_percentage_error`(...)	Mean absolute percentage error regression loss.
`metrics.r2_score`(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
`metrics.mean_poisson_deviance`(y_true, y_pred, *)	Mean Poisson deviance regression loss.
`metrics.mean_gamma_deviance`(y_true, y_pred, *)	Mean Gamma deviance regression loss.
`metrics.mean_tweedie_deviance`(y_true, y_pred, *)	Mean Tweedie deviance regression loss.

```
mape_train = round(100*mean_absolute_percentage_error(y_train, y_train_pred),1)
mape_test = round(100*mean_absolute_percentage_error(y_test, y_test_pred),1)
```

$$MAPE = \frac{100\%}{n} \sum \left| \frac{y - \hat{y}}{y} \right|$$

Multiplying by 100% converts to percentage

The residual

Each residual is scaled against the actual value

Step 3

Etapa 4

Benchmark moldel

Lineal regressor

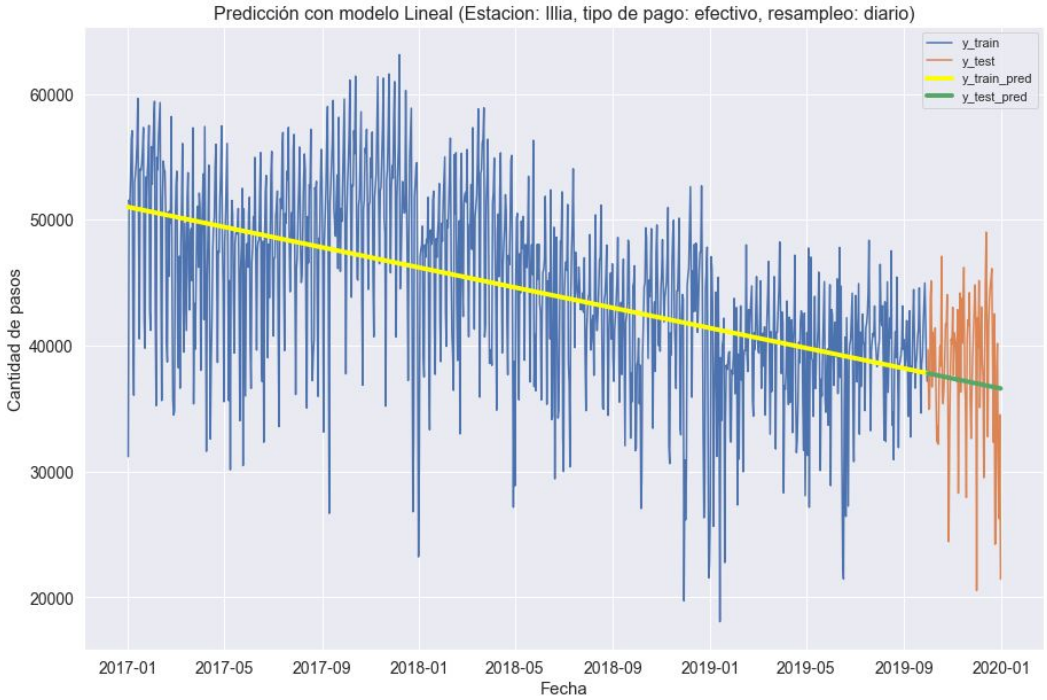
```
# Modelo de benchmark
from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()
```

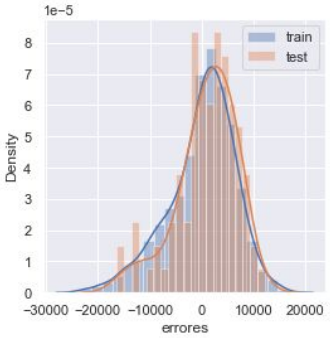
```
lr.fit(X_train, y_train)
```

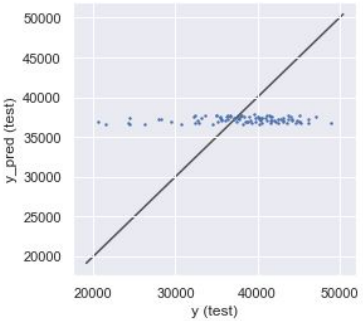
```
LinearRegression()
```

```
y_train_pred = lr.predict(X_train)
y_test_pred= lr.predict(X_test)
```




```
El modelo a evaluar:  Lineal regressor
Raíz del error cuadrático medio en Train: 6451.3
Raíz del error cuadrático medio en Test: 5825.3
Porcentaje del Error Absoluto medio en Train: 12.3
Porcentaje del Error Absoluto medio en Test: 13.2
```





Step 3

XGBoost

Único parametro

Etapa 4

```
X_train = df_train.fecha_num.values.reshape(-1, 1)
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.fecha_num.values.reshape(-1, 1)
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = reg.predict(X_train)
y_test_pred= reg.predict(X_test)
```

```
# Importamos la librería que nos permitirá hacer uso del modelo
import xgboost as xgb
```

```
# Instanciaremos el modelo con sus caracteristicas por defecto
reg = xgb.XGBRegressor()
```

```
# Corroboramos los hiperparametros que el modelo presenta
reg
```

```
XGBRegressor(base_score=None, booster=None, colsample_bylevel=None,
             colsample_bynode=None, colsample_bytree=None, gamma=None,
             gpu_id=None, importance_type='gain', interaction_constraints=None,
             learning_rate=None, max_delta_step=None, max_depth=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             random_state=None, reg_alpha=None, reg_lambda=None,
             scale_pos_weight=None, subsample=None, tree_method=None,
             validate_parameters=None, verbosity=None)
```

```
# Hacemos que el modelo se entrene con el set de entrenamiento (X_train, y_train)
# Caracteristicas:
# evalset : el modelo evaluara las métricas mape y rmse y se detendra tras 50 iteraciones
# en las que el modelo ho haya logrado obtener una metrica más baja.
# evalmetric: serán las metricas que emplearemos para evaluar lo pasos del del entrenamiento.
# La última de ellas es la que se constituye como criterio de parada para elearly stopping.
# early_stopping_rounds: cantidad de rondas que avanzará sin (si incluimos este parameto se almacenará el mejor modelo respecto a la metrica de evaluación)
# (Sino el modelo evaluará hasta el valor n_estimators y tomnara el valor del ultimo estimador)
# Verbose: Imprimirá cada una de los pasos del entrenamiento con sus respectivas evaluaciones sobre el eval_set
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        eval_metric= ['mape','rmse'],
        early_stopping_rounds=50,
        verbose=1
       )
```

```
[0]   validation_0-mape:0.69327   validation_0-rmse:31883.75781   validation_1-mape:0.67426   validation_1-rmse:26736.20508
[1]   validation_0-mape:0.47958   validation_0-rmse:22783.76562   validation_1-mape:0.45826   validation_1-rmse:18988.77930
[2]   validation_0-mape:0.33406   validation_0-rmse:16550.91602   validation_1-mape:0.31772   validation_1-rmse:13634.20996
[3]   validation_0-mape:0.23911   validation_0-rmse:12313.71094   validation_1-mape:0.23537   validation_1-rmse:10175.92578
[4]   validation_0-mape:0.18064   validation_0-rmse:9507.04004    validation_1-mape:0.18421   validation_1-rmse:8038.71582
[5]   validation_0-mape:0.14765   validation_0-rmse:7708.86377    validation_1-mape:0.15717   validation_1-rmse:6858.51123
[6]   validation_0-mape:0.12707   validation_0-rmse:6591.61670    validation_1-mape:0.14139   validation_1-rmse:6219.93848
[7]   validation_0-mape:0.11443   validation_0-rmse:5918.63379    validation_1-mape:0.13275   validation_1-rmse:5857.66162
[8]   validation_0-mape:0.10632   validation_0-rmse:5517.34375    validation_1-mape:0.13044   validation_1-rmse:5775.66943
[9]   validation_0-mape:0.10003   validation_0-rmse:5253.71875    validation_1-mape:0.13048   validation_1-rmse:5776.99609
[10]  validation_0-mape:0.09685   validation_0-rmse:5111.08740    validation_1-mape:0.13028   validation_1-rmse:5766.85303
```

Step 3

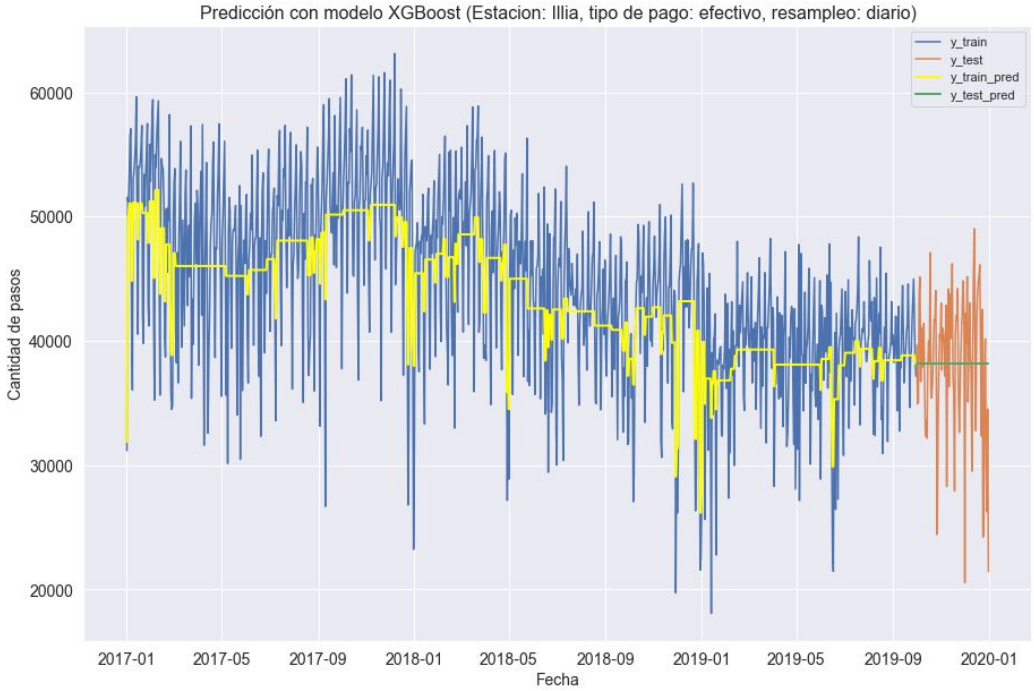
XGBoost

Único parametro

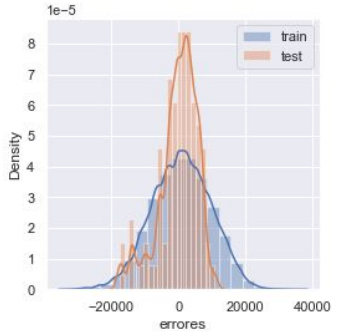
Etapa 4

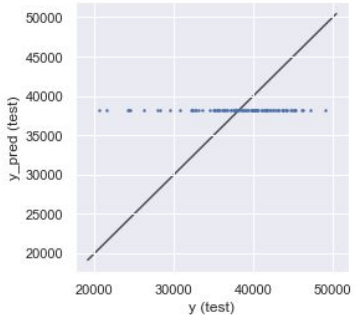
```
X_train = df_train.fecha_num.values.reshape(-1, 1)
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.fecha_num.values.reshape(-1, 1)
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = reg.predict(X_train)
y_test_pred= reg.predict(X_test)
```




```
El modelo a evaluar:  XGBoost
Raíz del error cuadrático medio en Train: 5111.1
Raíz del error cuadrático medio en Test: 5766.9
Porcentaje del Error Absoluto medio en Train: 9.7
Porcentaje del Error Absoluto medio en Test: 13.0
```





Step 3

Etapa 4

XGBoost

Múltiples parametros

```
X_train = df_train.drop(columns='cantidad_pasos')
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.drop(columns='cantidad_pasos')
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = reg.predict(X_train)
y_test_pred= reg.predict(X_test)
```

```
import xgboost as xgb
from xgboost import plot_importance, plot_tree
```

```
reg = xgb.XGBRegressor()
reg
```

```
XGBRegressor(base_score=None, booster=None, colsample_bylevel=None,
             colsample_bynode=None, colsample_bytree=None, gamma=None,
             gpu_id=None, importance_type='gain', interaction_constraints=None,
             learning_rate=None, max_delta_step=None, max_depth=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             random_state=None, reg_alpha=None, reg_lambda=None,
             scale_pos_weight=None, subsample=None, tree_method=None,
             validate_parameters=None, verbosity=None)
```

```
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        early_stopping_rounds=50,
        eval_metric= ['mape','rmse'],
        verbose=True)
```

```
[0]    validation_0-mape:0.69672    validation_0-rmse:31819.18555    validation_1-mape:0.68830    validation_1-rmse:27191.35938
[1]    validation_0-mape:0.48409    validation_0-rmse:22617.60938    validation_1-mape:0.46090    validation_1-rmse:18920.53906
[2]    validation_0-mape:0.33845    validation_0-rmse:16250.85449    validation_1-mape:0.31597    validation_1-rmse:13461.04492
[3]    validation_0-mape:0.24293    validation_0-rmse:11874.39062    validation_1-mape:0.23019    validation_1-rmse:9838.56055
[4]    validation_0-mape:0.17852    validation_0-rmse:8859.53418     validation_1-mape:0.17744    validation_1-rmse:7574.64941
[5]    validation_0-mape:0.13684    validation_0-rmse:6854.20312     validation_1-mape:0.14689    validation_1-rmse:6362.16211
[6]    validation_0-mape:0.10988    validation_0-rmse:5570.06250     validation_1-mape:0.12992    validation_1-rmse:5675.70215
[7]    validation_0-mape:0.09126    validation_0-rmse:4684.71728     validation_1-mape:0.12086    validation_1-rmse:5357.02734
[8]    validation_0-mape:0.07968    validation_0-rmse:4153.39111     validation_1-mape:0.11818    validation_1-rmse:5308.08887
[9]    validation_0-mape:0.07170    validation_0-rmse:3792.49145     validation_1-mape:0.11800    validation_1-rmse:5292.42578
[10]   validation_0-mape:0.06665    validation_0-rmse:3560.74707     validation_1-mape:0.11846    validation_1-rmse:5326.15527
[11]   validation_0-mape:0.06353    validation_0-rmse:3432.36865     validation_1-mape:0.11897    validation_1-rmse:5371.30322
[12]   validation_0-mape:0.06080    validation_0-rmse:3308.42456     validation_1-mape:0.11923    validation_1-rmse:5403.81494
[13]   validation_0-mape:0.05828    validation_0-rmse:3195.09644     validation_1-mape:0.11971    validation_1-rmse:5433.05420
[14]   validation_0-mape:0.05625    validation_0-rmse:3088.71509     validation_1-mape:0.12673    validation_1-rmse:5605.02930
[15]   validation_0-mape:0.05530    validation_0-rmse:3054.30298     validation_1-mape:0.12648    validation_1-rmse:5589.71289
[16]   validation_0-mape:0.05393    validation_0-rmse:2974.60742     validation_1-mape:0.12625    validation_1-rmse:5588.57178
[17]   validation_0-mape:0.05295    validation_0-rmse:2932.49609     validation_1-mape:0.12642    validation_1-rmse:5599.01221
[18]   validation_0-mape:0.05241    validation_0-rmse:2911.72070     validation_1-mape:0.12620    validation_1-rmse:5585.98047
[19]   validation_0-mape:0.05198    validation_0-rmse:2888.10010     validation_1-mape:0.12623    validation_1-rmse:5587.91699
[20]   validation_0-mape:0.05055    validation_0-rmse:2826.68066     validation_1-mape:0.12638    validation_1-rmse:5596.03516
[21]   validation_0-mape:0.04882    validation_0-rmse:2738.89917     validation_1-mape:0.12634    validation_1-rmse:5593.11084
[22]   validation_0-mape:0.04767    validation_0-rmse:2678.81006     validation_1-mape:0.12643    validation_1-rmse:5598.30176
[23]   validation_0-mape:0.04689    validation_0-rmse:2637.30591     validation_1-mape:0.13091    validation_1-rmse:5765.99316
[24]   validation_0-mape:0.04660    validation_0-rmse:2619.70874     validation_1-mape:0.13091    validation_1-rmse:5765.99902
[25]   validation_0-mape:0.04634    validation_0-rmse:2607.73193     validation_1-mape:0.13086    validation_1-rmse:5759.73096
```

Step 3

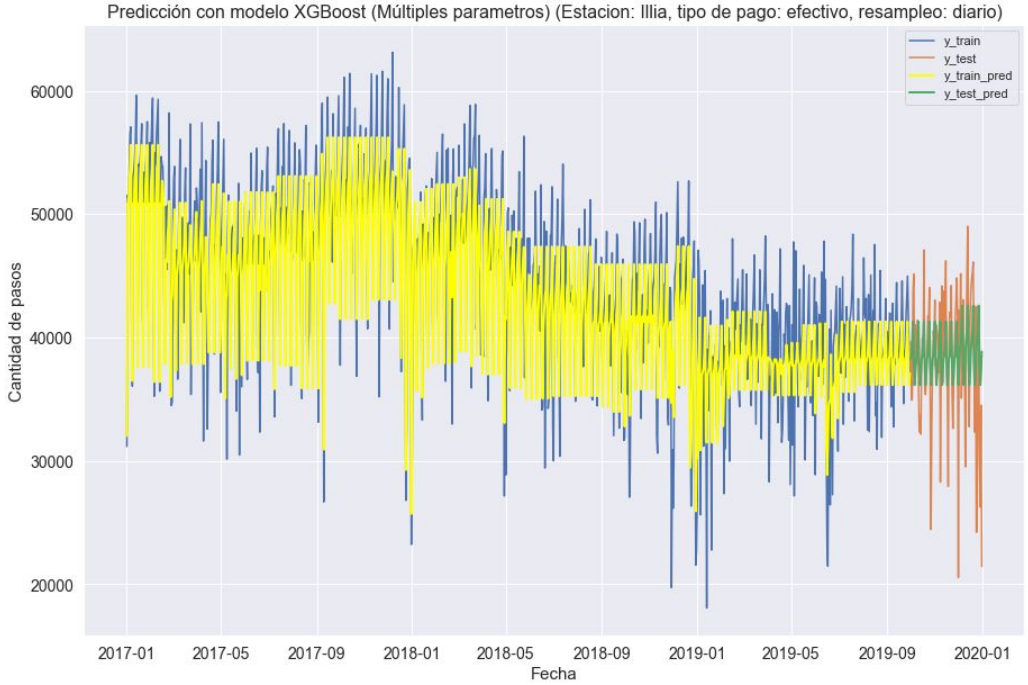
XGBoost

Múltiples parametros

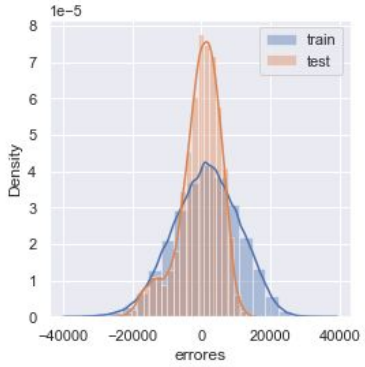
Etapa 4

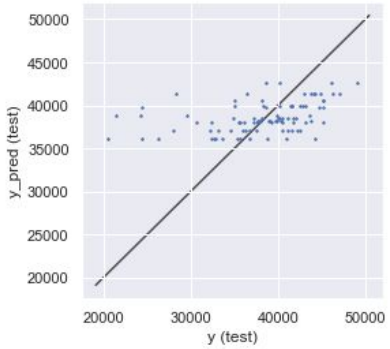
```
X_train = df_train.fecha_num.values.reshape(-1, 1)
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.fecha_num.values.reshape(-1, 1)
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = reg.predict(X_train)
y_test_pred= reg.predict(X_test)
```




```
El modelo a evaluar:  XGBoost (Múltiples parametros)
Raíz del error cuadrático medio en Train: 3792.5
Raíz del error cuadrático medio en Test: 5292.4
Porcentaje del Error Absoluto medio en Train: 7.2
Porcentaje del Error Absoluto medio en Test: 11.8
```



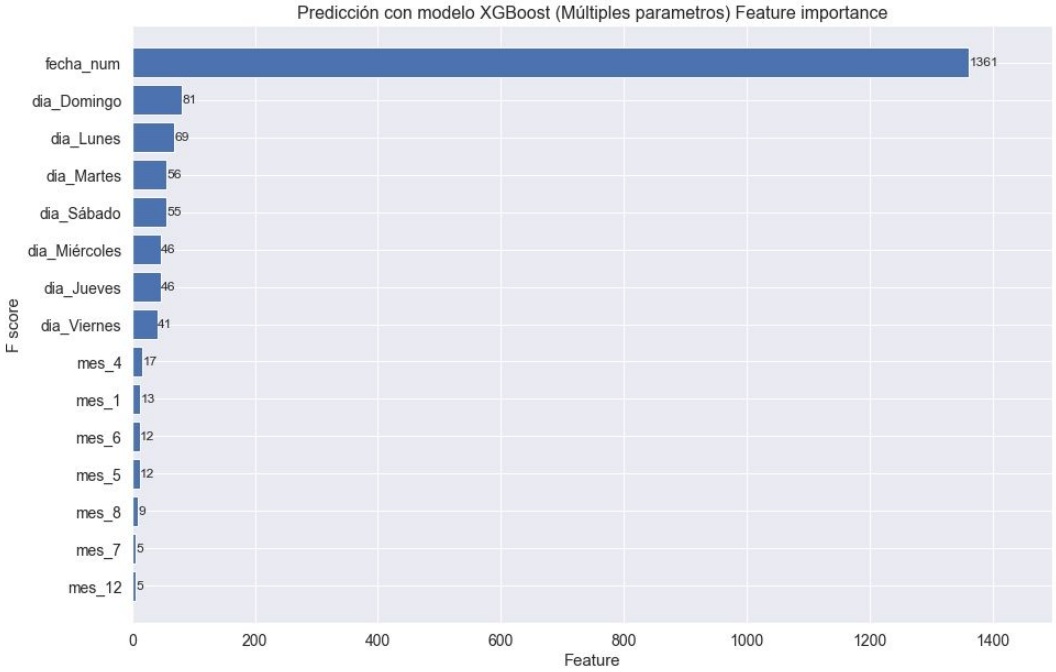


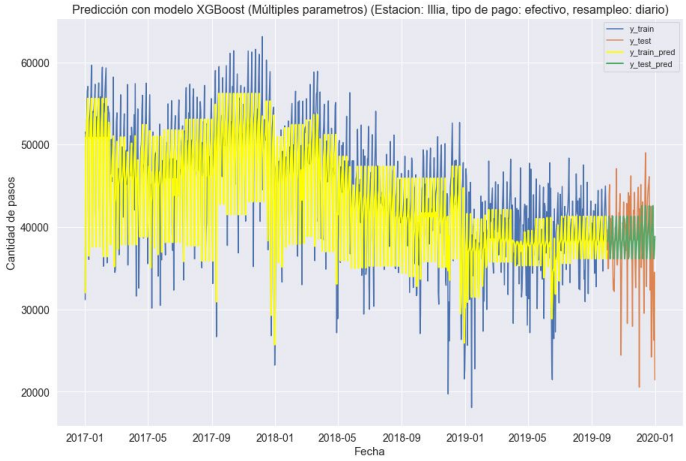
Step 3

XGBoost

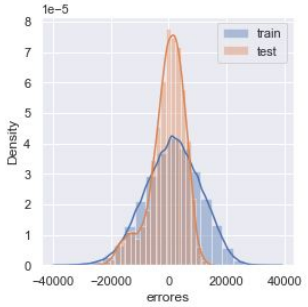
Múltiples parametros

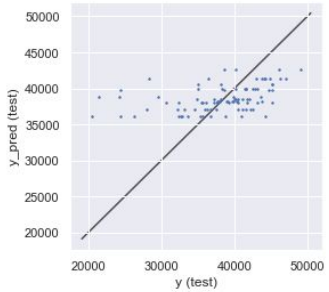
Etapa 4






```
El modelo a evaluar:  XGBoost (Múltiples parametros)
Raíz del error cuadrático medio en Train: 3792.5
Raíz del error cuadrático medio en Test: 5292.4
Porcentaje del Error Absoluto medio en Train: 7.2
Porcentaje del Error Absoluto medio en Test: 11.8
```





Step 4

Etapa 4

XGBoost

Múltiples parametros (Optimizado)

```
X_train = df_train.drop(columns='cantidad_pasos')
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.drop(columns='cantidad_pasos')
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = reg.predict(X_train)
y_test_pred= reg.predict(X_test)
```

```
import xgboost as xgb
from xgboost import plot_importance, plot_tree
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
```

```
xgb_model=xgb.XGBRegressor()
```

```
param_rs = {'nthread':[1, 2, 3, 4, 5],
            'objective':['reg:squarederror'],
            'learning_rate': np.arange(.05, 1, .05),
            'max_depth': np.arange(1,10),
            'min_child_weight': [3, 4, 5, 6, 7, 8],
            'subsample': np.arange(.05, 1, .05),
            'colsample_bytree': np.arange(.05, 1, .05),
            'n_estimators': [200, 300, 400, 500, 600]}
param_rs
```

```
{'nthread': [1, 2, 3, 4, 5],
 'objective': ['reg:squarederror'],
 'learning_rate': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 , 0.55,
        0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95]),
 'max_depth': array([1, 2, 3, 4, 5, 6, 7, 8, 9]),
 'min_child_weight': [3, 4, 5, 6, 7, 8],
 'subsample': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 , 0.55,
        0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95]),
 'colsample_bytree': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 , 0.55,
        0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95]),
 'n_estimators': [200, 300, 400, 500, 600]}
```

```
xgb_rs = RandomizedSearchCV(xgb_model, param_rs,n_iter=50, random_state=0, cv=5,
                            scoring='neg_root_mean_squared_error', verbose=True, n_jobs=-1, refit=True)
```

```
xgb_rs.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        early_stopping_rounds=50,
        verbose=True)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[0]     validation_0-rmse:16842.12695   validation_1-rmse:14255.70019
[1]     validation_0-rmse:7683.47607    validation_1-rmse:6927.71777
[2]     validation_0-rmse:5161.58008    validation_1-rmse:5276.70264
[3]     validation_0-rmse:4224.71680    validation_1-rmse:5375.77441
```

Step 4

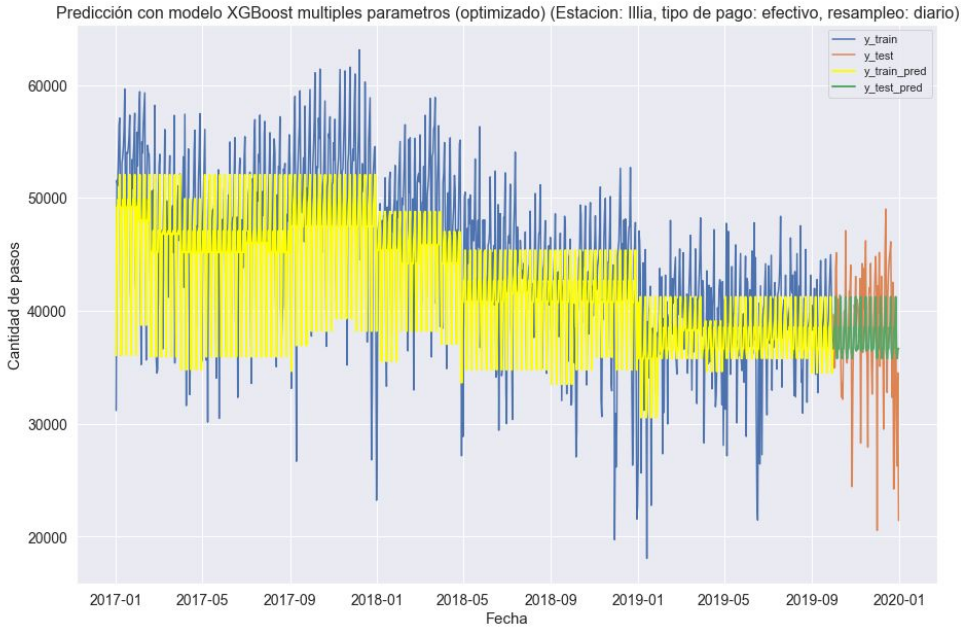
Etapa 4

XGBoost

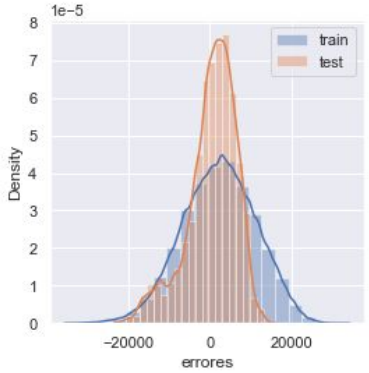
Múltiples parametros (Optimizado)

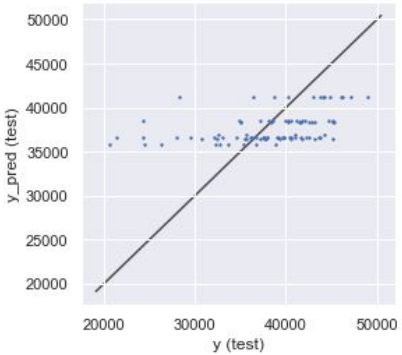
```
X_train = df_train.fecha_num.values.reshape(-1, 1)
y_train = df_train.cantidad_pasos.values.reshape(-1, 1)
X_test = df_test.fecha_num.values.reshape(-1, 1)
y_test = df_test.cantidad_pasos.values.reshape(-1, 1)
```

```
y_train_pred = xgb_rs.predict(X_train)
y_test_pred= xgb_rs.predict(X_test)
```




```
El modelo a evaluar:  XGBoost multiples parametros optimizado
Raíz del error cuadrático medio en Train: 5161.6
Raíz del error cuadrático medio en Test: 5276.7
Porcentaje del Error Absoluto medio en Train: 9.9
Porcentaje del Error Absoluto medio en Test: 12.0
```



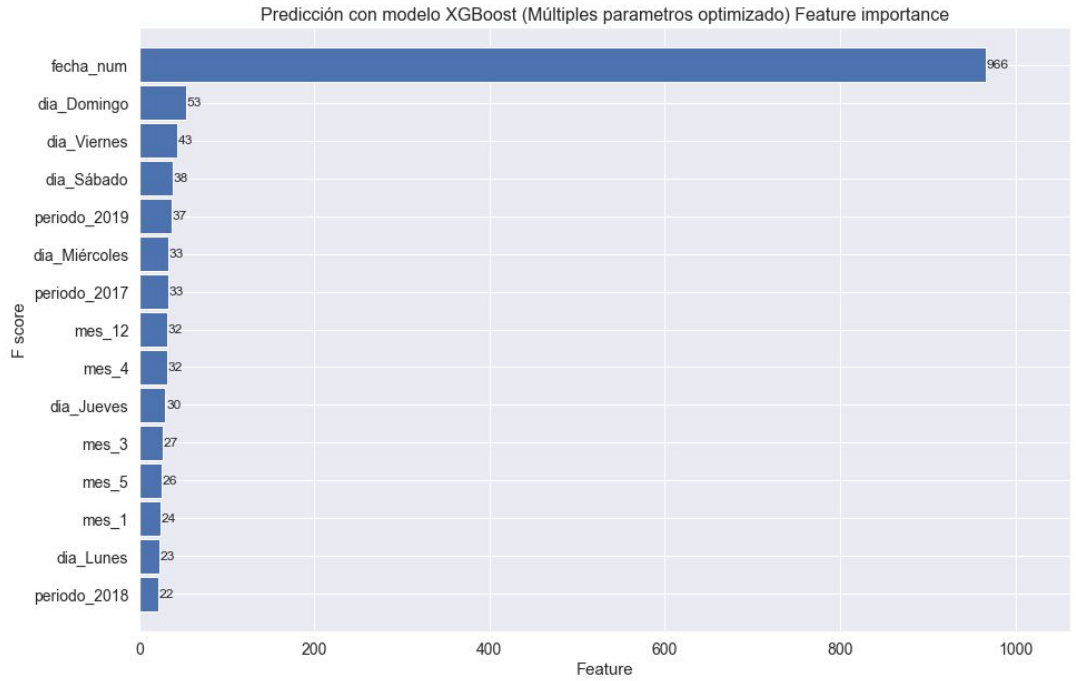


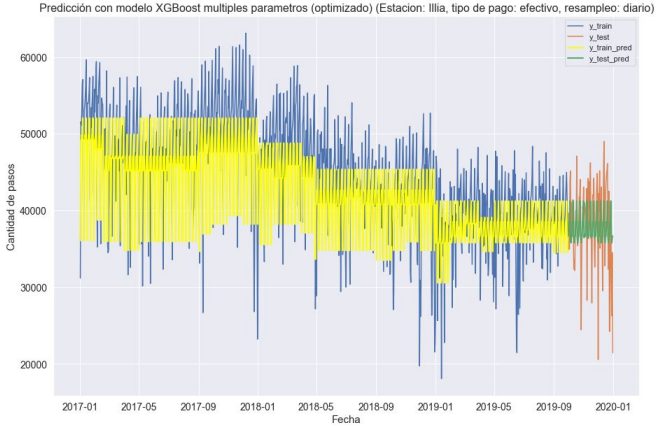
Step 4

Etapa 4

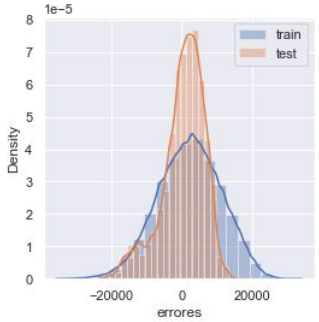
XGBoost

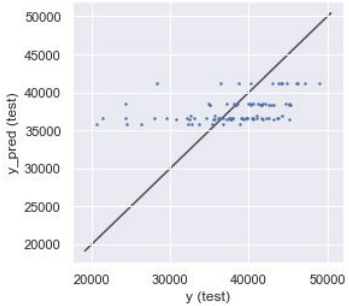
Múltiples parametros (Optimizado)






```
El modelo a evaluar:  XGBoost multiples parametros optimizado
Raíz del error cuadrático medio en Train: 5161.6
Raíz del error cuadrático medio en Test: 5276.7
Porcentaje del Error Absoluto medio en Train: 9.9
Porcentaje del Error Absoluto medio en Test: 12.0
```





Step 5

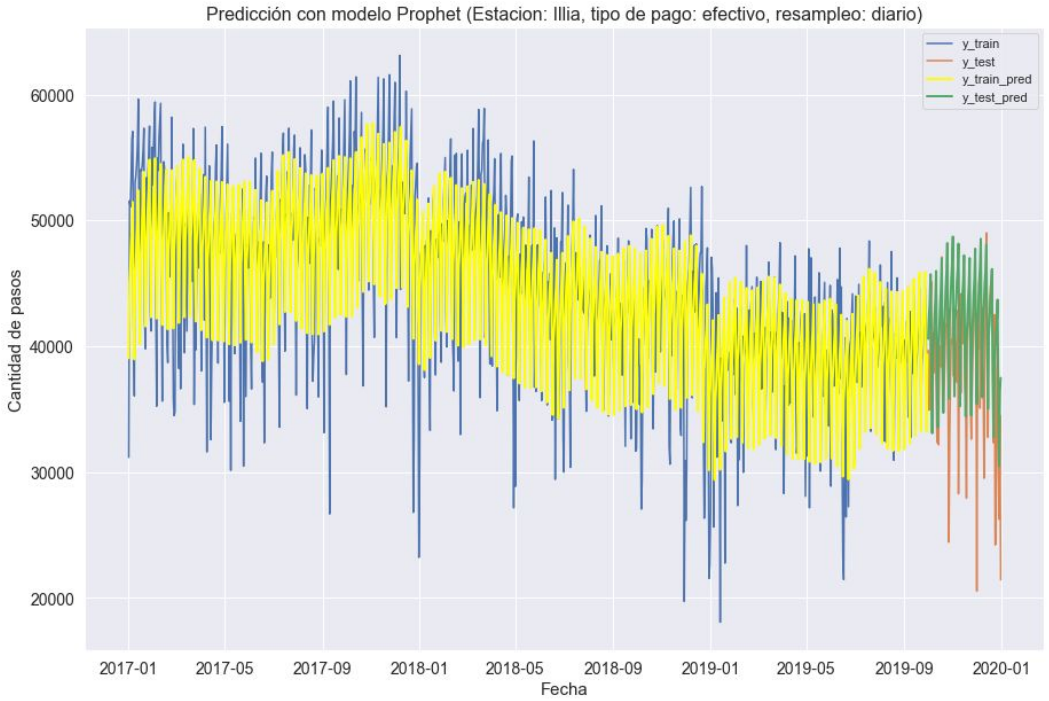
Prophet

Múltiples parametros

Etapa 4

```
# Importamos la librería
from fbprophet import Prophet

# Instanciamos el modelo
prophet_model = Prophet()

# Entrenamos nuestro modelo con los datos
prophet_model.fit(df_train_prophet)
```

```
y_train_pred = df_train_prophet_pred.yhat.values.reshape(-1, 1)
y_test_pred= df_test_prophet_pred.yhat.values.reshape(-1, 1)
```



df_train_prophet

	ds	y
0	2017-01-01	31183
1	2017-01-02	51568
2	2017-01-03	51093
3	2017-01-04	53332
4	2017-01-05	56486
...	...	...
998	2019-09-26	43110
999	2019-09-27	44985
1000	2019-09-28	41925
1001	2019-09-29	38348
1002	2019-09-30	37170

1003 rows × 2 columns

df_test_prophet

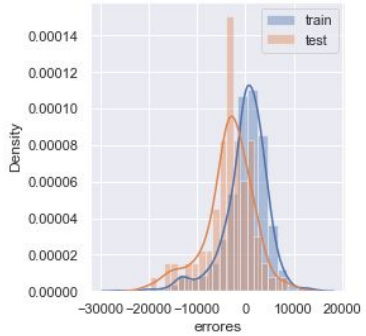
	ds	y
0	2019-10-01	39669
1	2019-10-02	34943
2	2019-10-03	35054
3	2019-10-04	43775
4	2019-10-05	45150
...	...	...
87	2019-12-27	40182
88	2019-12-28	30681
89	2019-12-29	26259
90	2019-12-30	34523
91	2019-12-31	21447

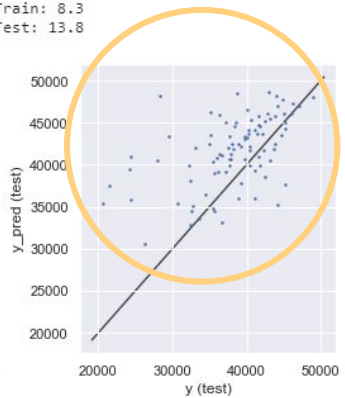
92 rows × 2 columns

df_train_prophet_pred

	ds	yhat
0	2017-01-01	39066.394756
1	2017-01-02	45111.617934
2	2017-01-03	46311.755100
3	2017-01-04	48109.759523
4	2017-01-05	49250.201655
...	...	...
998	2019-09-26	43724.241317
999	2019-09-27	45895.909636
1000	2019-09-28	37826.241508
1001	2019-09-29	33214.536879
1002	2019-09-30	39348.869595

```
El modelo a evaluar:  Prophet
Raíz del error cuadrático medio en Train: 4663.9
Raíz del error cuadrático medio en Test: 6110.3
Porcentaje del Error Absoluto medio en Train: 8.3
Porcentaje del Error Absoluto medio en Test: 13.8
```





Etapa 5

Predicciones para 2020 y propuestas por explorar

Step 1

Predicciones para 2020

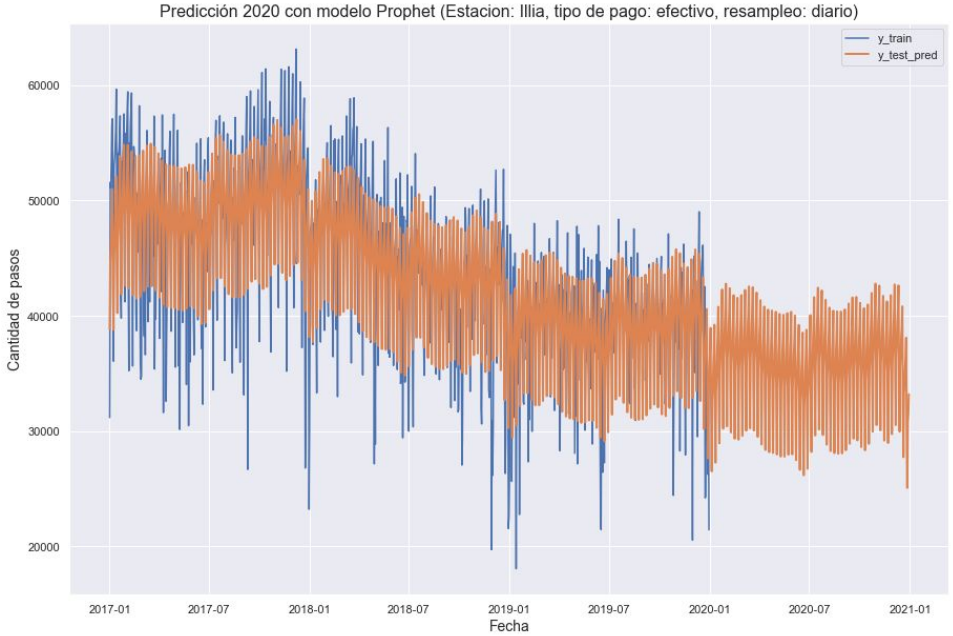
Prophet

Etapa 5

```
# Importamos la librería
from fbprophet import Prophet
```

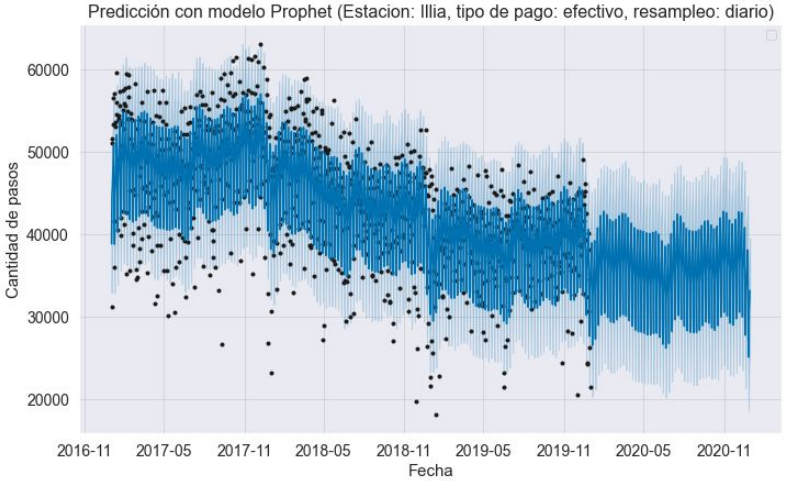
```
# Instanciamos el modelo
prophet_model = Prophet()
```

```
prophet_model.fit(df_train_prophet_forecast)
```




```
future = prophet_model.make_future_dataframe(periods=365)
```

```
forecast = prophet_model.predict(future)
```



Step 2

Etapa 5

Propuestas por explorar

towards data science

Upgrade

Follow 543K Followers · Editors' Picks Features Explore Contribute About

Multivariable time series forecasting using Stateless Neural Networks

Arjeus A. Guevarra Aug 27, 2019 · 12 min read ★

Forecasting with multiple variables that aids a target variable with stateless deep learning.

. . .

towards data science

Upgrade

Follow 543K Followers · Editors' Picks Features Explore Contribute About

LSTM for time series prediction

Roman Orac Sep 27, 2019 · 11 min read ★

towards data science

Upgrade

Follow 543K Followers · Editors' Picks Features Explore Contribute About

Welcome back. You are signed in as **alexandergribenchenko@gmail.com**. Not you?

Deep Learning for Time Series Classification (InceptionTime)

New Deep Learning (GoogleNet-like) model for Time Series Classification.

Vasilis Stylianou, PhD Jan 21, 2020 · 9 min read ★

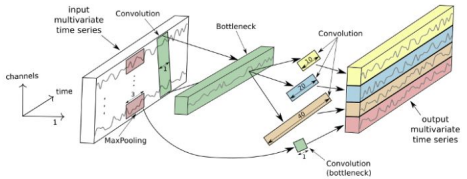


Fig. 1: The Inception module of InceptionTime.

github.com/alexandergribenchenko/Acamica_Time_Series_PY_03

¡GRACIAS!

ACÁMICA

Carrera Data Science