Variable Cepheid classification from unprocessed light curves

Maximiliano Garavito Chtefan

Justificación

Las cefeidas clásicas [también conocidas como estrellas delta Cepheida, cefeidas tipo I o cefeidas de población I] se encuentran entre las estrellas variables más famosas e importantes.

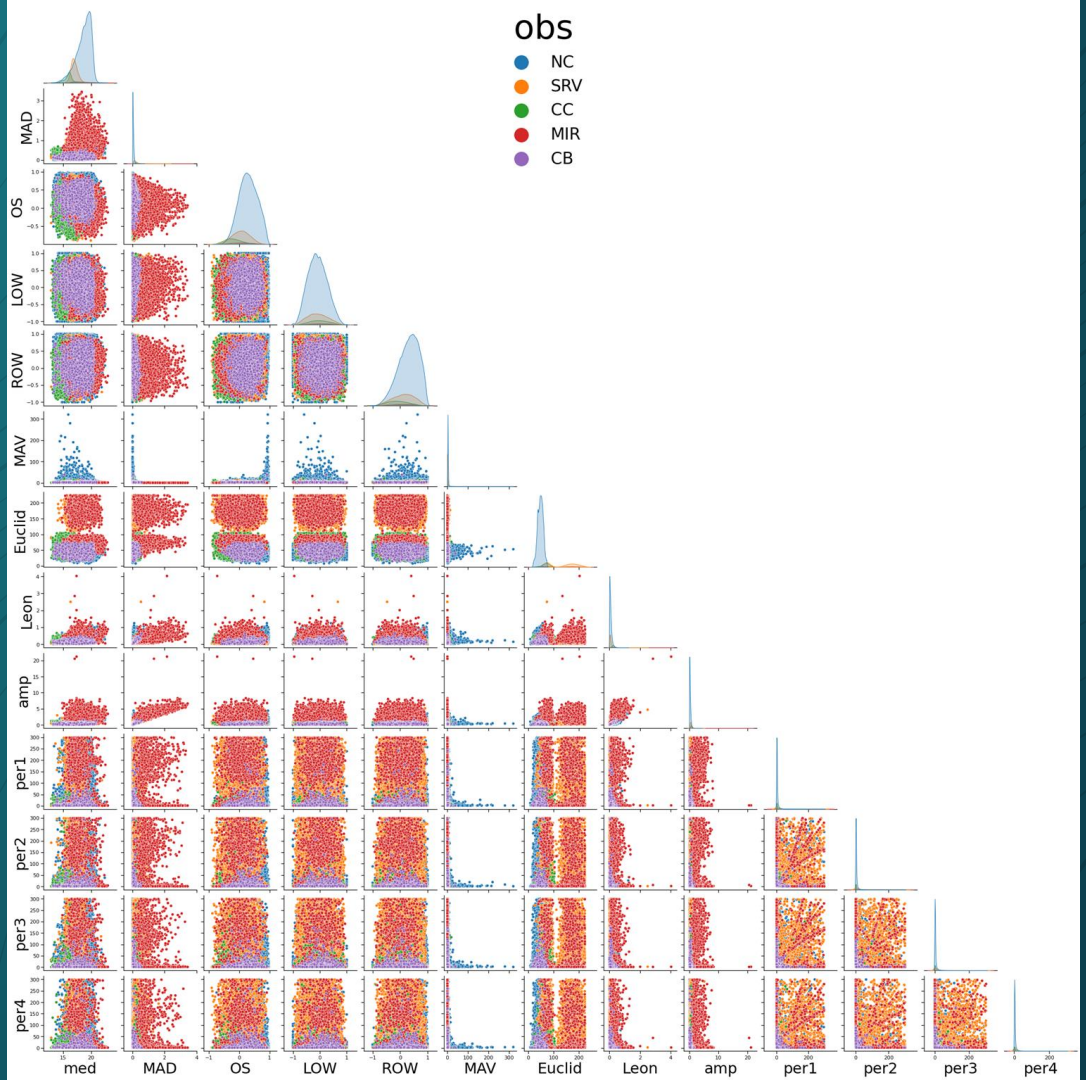
Son estrellas relativamente jóvenes, masivas, de pulsaciones radiales con relaciones bien definidas entre la magnitud absoluta y el período, lo que las convierte en importantes indicadores de distancias intra y extragálaticas.

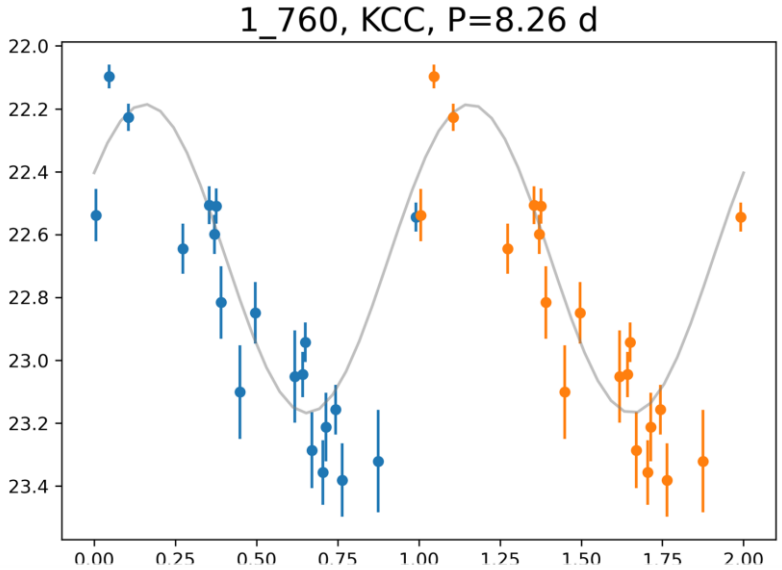
Justificación

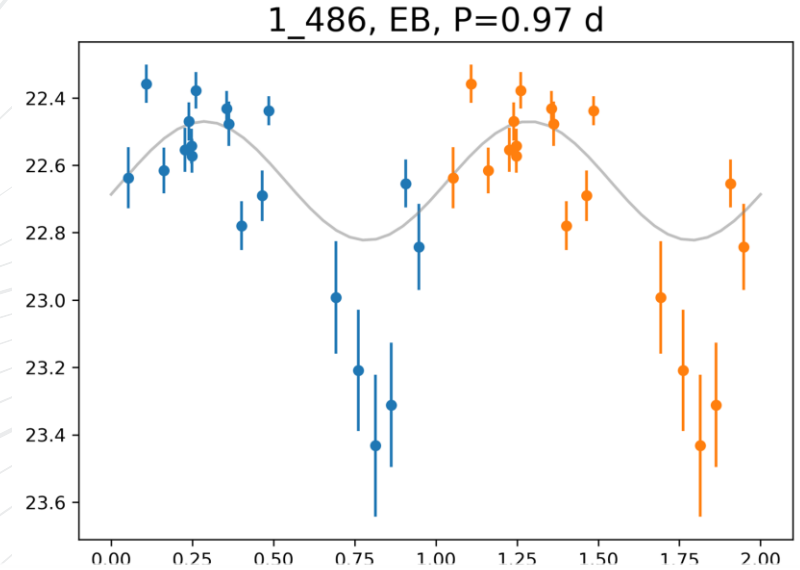
Sin embargo, clasificar estas estrellas puede ser un problema complejo, que requiere de mucha interacción con expertos.

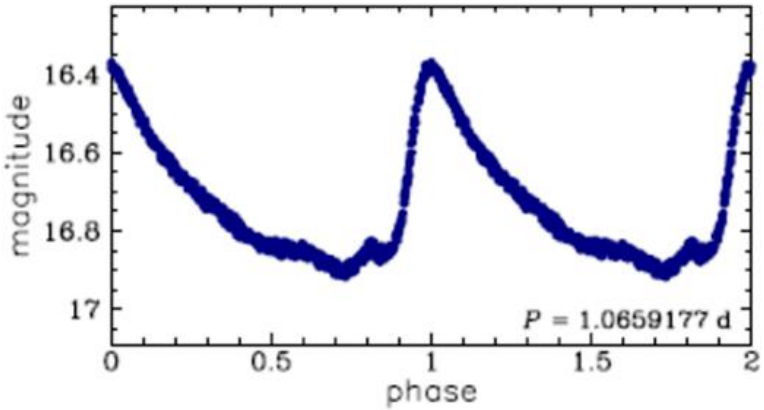
Esto es así debido a que no es sencillo diferenciar cefeidas variables de otros tipos de estrellas como las RR Lyrae variable. Además, es común que hayan curvas de luz de muy pocas noches de observación en periodos muy largos de tiempo.

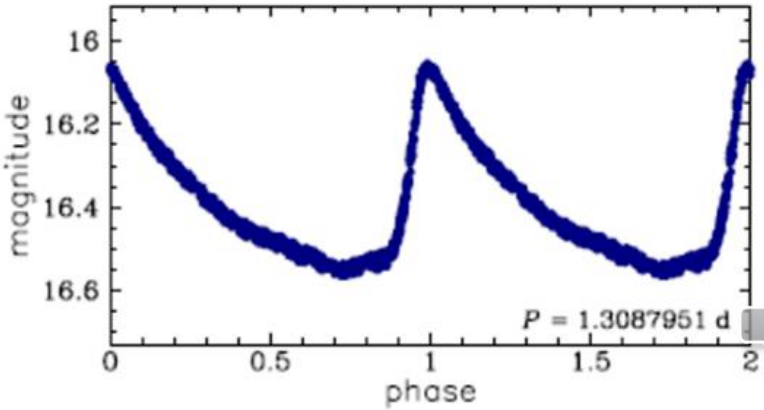
Esto se traduce en que cada curva de luz deba ser individualmente analizada con heurísticas de carácter estadístico, para así conocer si es una posible candidata a cefeida variable y posteriormente simular los periodos más probables; finalmente determinanando si es cefeida variable o no. Lo que requiere de un trabajo extenso y dispendioso por parte de astrónomos.


obs
NC
SRV
CC
MIR
CB
med
MAD
OS
LOW
ROW
MAV
Euclid
Leon
amp
per1
per2
per3
per4


1_760, KCC, P=8.26 d
22.0
22.2
22.4
22.6
22.8
23.0
23.2
23.4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

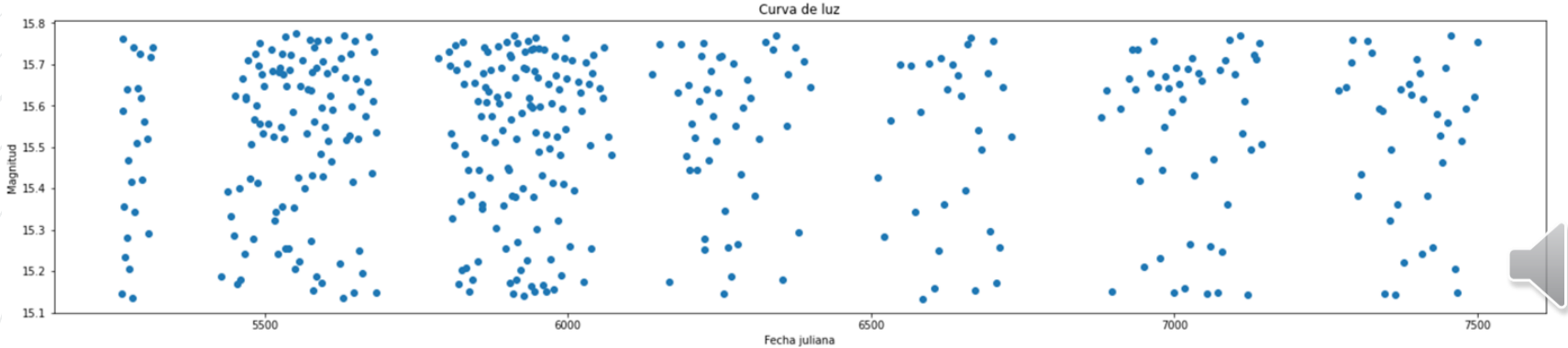

1_486, EB, P=0.97 d
22.4
22.6
22.8
23.0
23.2
23.4
23.6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

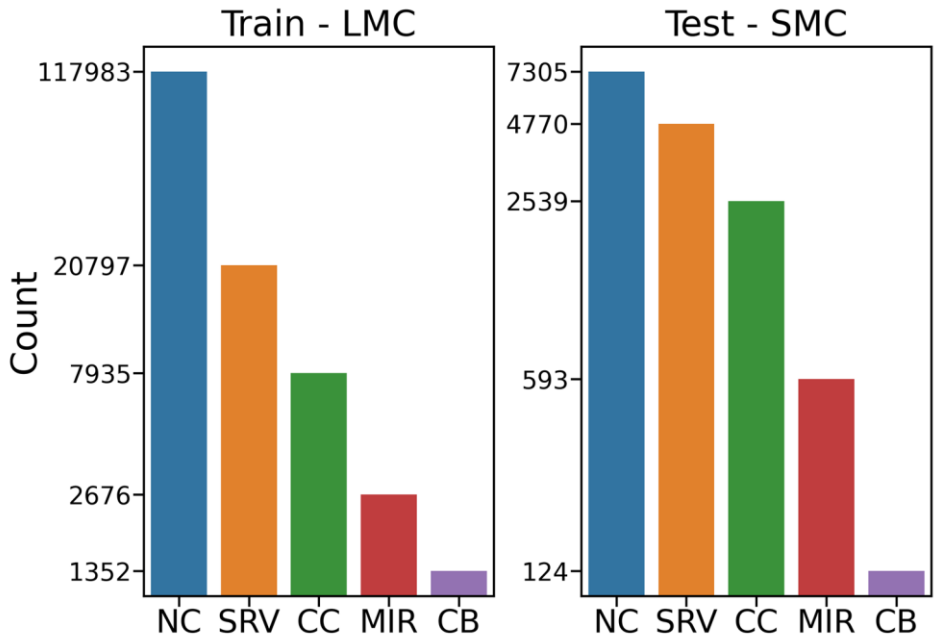

magnitude
16.4
16.6
16.8
17
P = 1.0659177 d
0
0.5
1
1.5
2
phase


magnitude
16
16.2
16.4
16.6
P = 1.3087951 d
0
0.5
1
1.5
2
phase

Datasets

OGLE Project: estudio del cielo a gran escala realizado desde 1992 por astrónomos asociados con el Observatorio Astronómico de la Universidad de Varsovia con el telescopio Observatorio Las Campanas, Chile. El Catálogo OGLE de Estrellas Variables consta de más de 400.000 objetos y ahora es el mayor conjunto de estrellas variables del mundo.





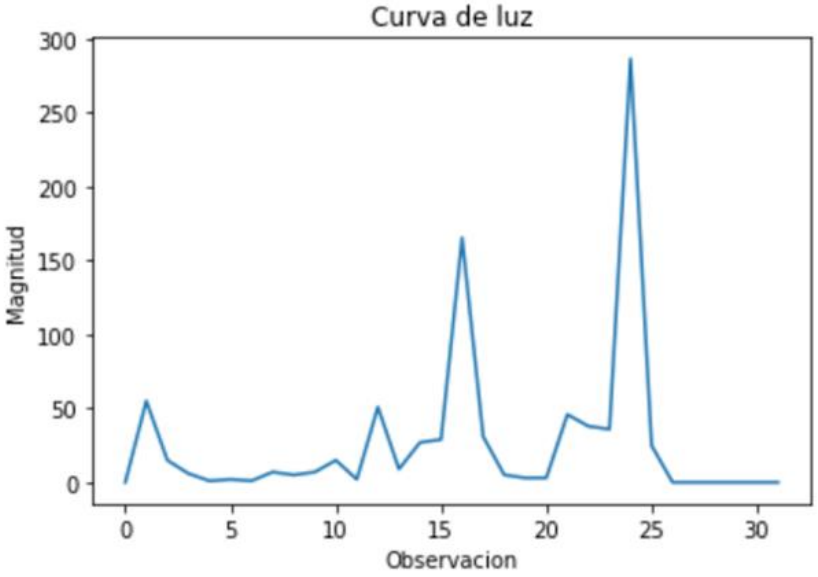
LMC: Large magellanic cloud
SMC: Small magellanic cloud

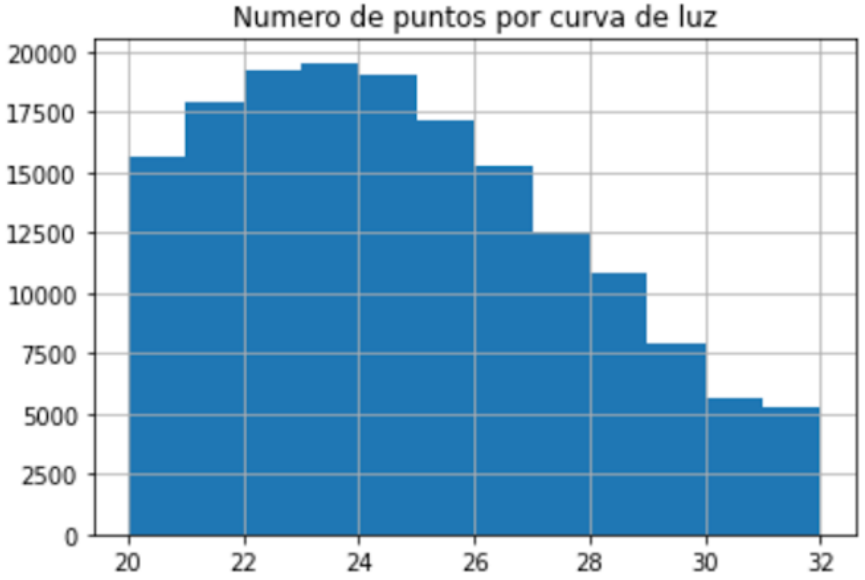
MIR: Mira- red giant
CC: Cepheid
NC: No contact
CB: Contact binary
SRV: Semiregular variable

Desbalance del dataset del proyecto OGLE por tipos de estrellas variables

Subsampling and data augmentation

```python
def subsample(serie,numbers,dist,mask_value):
  '''Function to decompose a time serie into multiple randomly subsampled time series with number of
  numbers with distribution of number of points given by dist, the subsampled time series are padded
  the return is a numpy array of shape (n,max_number,channels) where n is the number of generated sub
  max_number is the greatest value of numbers and channels is the features of the serie'''
  max_steps=np.max(numbers)
  l=len(serie)
  if l<=max_steps:
    arr=np.ones((1,max_steps,3))*mask_value
    s=np.shape(serie)
    arr[0,:s[0]]=serie
    return arr
  else:
    n=np.random.choice(numbers,replace=False, p=dist)
    sel=np.sort(np.random.choice(l,n,replace=False))
    serie_trunc=serie[sel]
    s=np.shape(serie_trunc)
    arr=np.ones((1,max_steps,3))*mask_value
    arr[0,:s[0]]=serie_trunc
    bool_sel=np.ones(l).astype(bool)
    bool_sel[sel]=False
    serie=serie[bool_sel]
    l=len(serie)
    while l>=max_steps:
      n=np.random.choice(numbers,replace=False, p=dist)
      sel=np.sort(np.random.choice(l,n,replace=False))
      serie_trunc=serie[sel]
      s=np.shape(serie_trunc)
      t=np.ones((1,max_steps,3))*mask_value
      t[0,:s[0]]=serie_trunc
      arr=np.concatenate((arr,t))
      bool_sel=np.ones(l).astype(bool)
      bool_sel[sel]=False
      serie=serie[bool_sel]
      l=len(serie)
    if l>=np.min(numbers):
      s=np.shape(serie)
      t=np.ones((1,max_steps,3))*mask_value
      t[0,:s[0]]=serie
      arr=np.concatenate((arr,t))
    return arr
```





Datos remuestreados para que coincidan con la colección Uniandes de Estrellas Variables (El telescopio MPG / ESO 2.2m)

Custom Layer

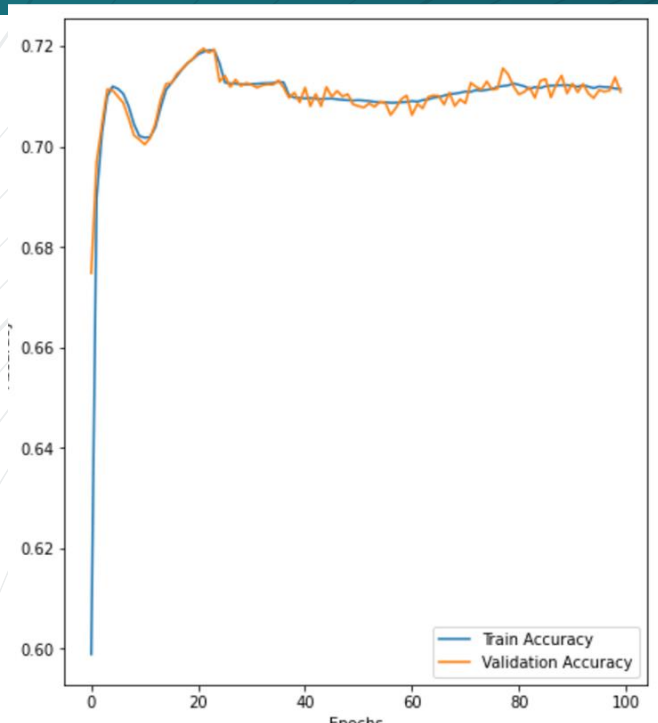
```
class lstm_bottleneck(tfkl.Layer):
    def __init__(self, lstm_units, time_steps, **kwargs):
        self.lstm_units = lstm_units
        self.time_steps = time_steps
        self.lstm_layer = tfkl.Bidirectional(tfkl.LSTM(lstm_units,return_sequences=False))
        self.repeat_layer = tfkl.RepeatVector(time_steps)
        super(lstm_bottleneck, self).__init__(**kwargs)

    def call(self, inputs):
        # just call the two initialized layers
        return self.repeat_layer(self.lstm_layer(inputs))

    def compute_mask(self, inputs, mask=None):
        # return the input_mask directly
        return mask
```

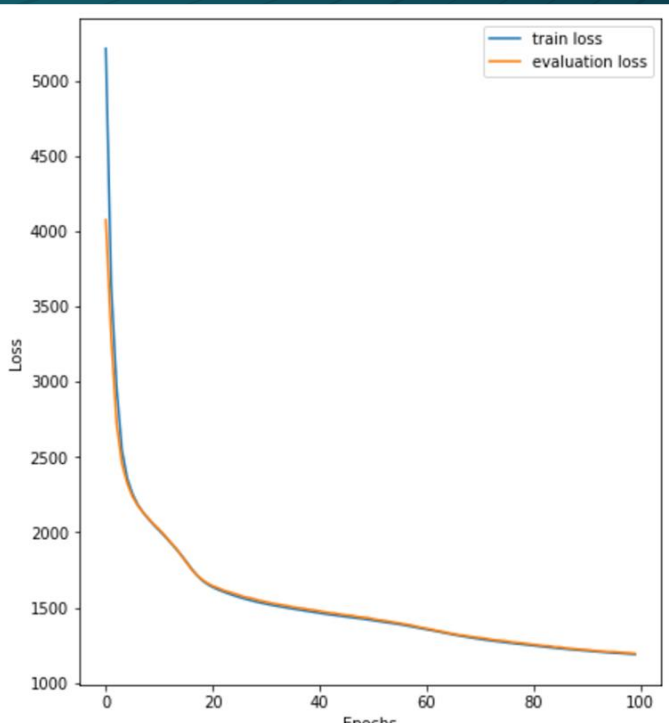
Arquitecturas implementadas

```
Layer (type)                 Output Shape              Param #
=================================================================
masking_1 (Masking)          (None, 32, 3)             0
_________________________________________________________________
bidirectional_4 (Bidirection (None, 32, 64)            9216
_________________________________________________________________
lstm_bottleneck_1 (lstm_bott (None, 32, 14)            4032
=================================================================
Total params: 13,248
Trainable params: 13,248
Non-trainable params: 0
```

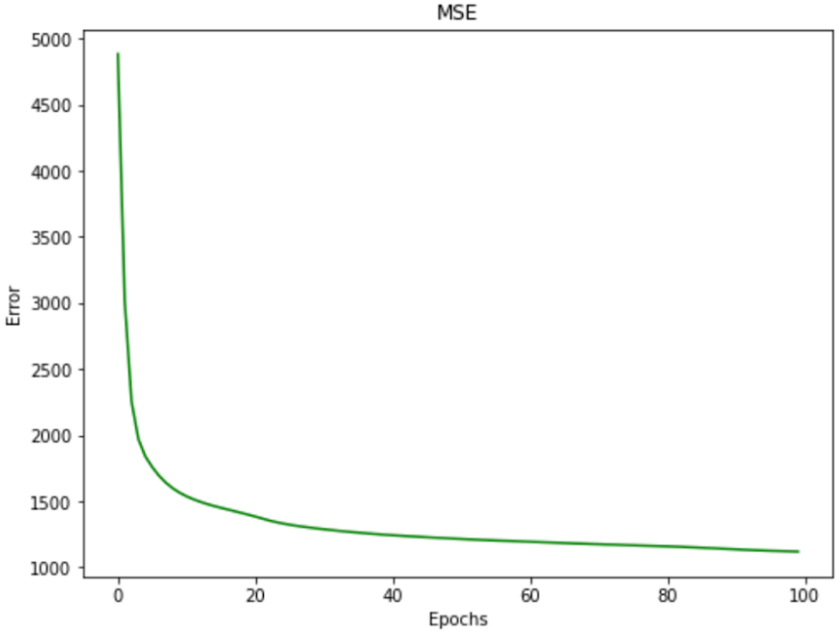
```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_6 (Bidirection (None, 32, 64)            12032
_________________________________________________________________
bidirectional_7 (Bidirection (None, 32, 6)             1632
_________________________________________________________________
dense_1 (Dense)              (None, 32, 3)             21
=================================================================
Total params: 13,685
Trainable params: 13,685
Non-trainable params: 0
```

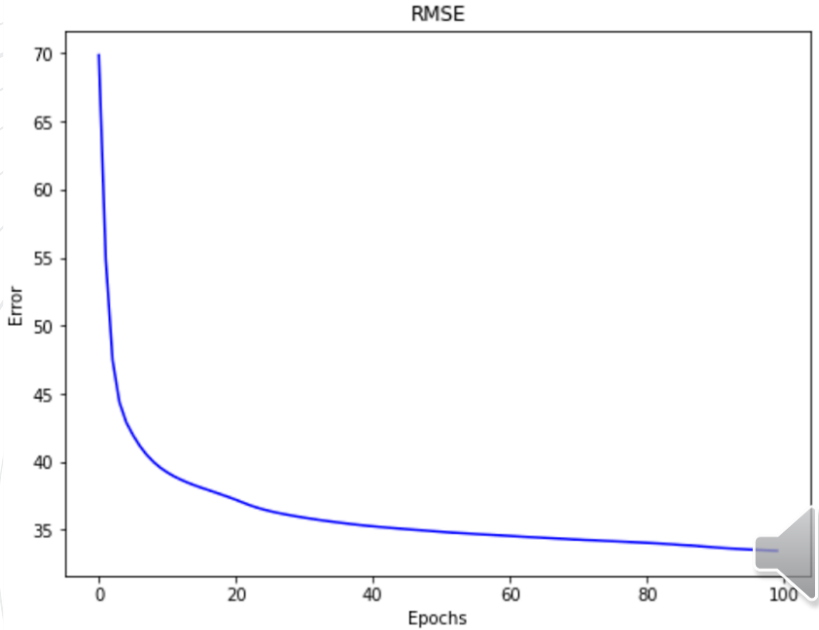
Resultados


0.72
0.70
0.68
0.66
0.64
0.62
0.60
0
20
40
60
80
100
Train Accuracy
Validation Accuracy


train loss
evaluation loss
Loss
5000
4500
4000
3500
3000
2500
2000
1500
1000
0
20
40
60
80
100

Resultados


MSE
Error
Epochs
5000
4500
4000
3500
3000
2500
2000
1500
1000
0
20
40
60
80
100


RMSE
Error
Epochs
70
65
60
55
50
45
40
35
0
20
40
60
80
100

Funciones de validación

```
def mask(arr,mask_value=-8):
    sel=np.diff(arr,axis=0)
    sel=np.sum(sel,axis=1)!=0.
    sel=np.concatenate((sel,[False]))
    return sum(sel)
```

```
def masked_mse(prediction,real_vals,mask,mask_value=-8):
    dif=prediction[:mask]-real_vals[:mask]
    mse = (np.square(dif)).mean(axis=None)
    return mse
```

MSE por clase

Succes!

```
Mean MSE for Cep data is:  5819.561664891817
Max MSE for Cep data is: 190481.8768829917
Min MSE for Cep data is: 23.640436702061876
```

```
Mean MSE for Rrlyr data is:  11966346.737544492
Max MSE for Rrlyr data is: 22383903.090269934
Min MSE for Rrlyr data is: 9820725.792896278
```

```
Mean MSE for Acep data is:  10483469.925480977
Max MSE for Acep data is: 11124888.48133775
Min MSE for Acep data is: 9914071.185982602
```

```
Mean MSE for T2CEP data is:  10483469.925480977
Max MSE for T2CEP data is: 11124888.48133775
Min MSE for T2CEP data is: 9914071.185982602
```

```
Mean MSE for Ecl data is:  10483469.925480977
Max MSE for Ecl data is: 11124888.48133775
Min MSE for Ecl data is: 9914071.185982602
```