SIMULACIÓN MATEMÁTICA

Modelado y predicción del tipo de cambio Dólar Estadounidense-Peso Mexicano mediante *Ajuste de Curvas*

ITESO, Universidad Jesuita de Guadalajara

745973

KEVIN EMILIANO AYALA MONTOYA

kevin.ayala@iteso.mx

objetivo general:

Desarrollar un modelo predictivo con ajuste de curvas utilizando datos históricos del tipo de cambio dólar-peso mexicano de los últimos 23 meses, con el fin de estimar el valor del tipo de cambio en el presente.

objetivos específicos:

1. Obtener y preprocesar los datos históricos del precio del dólar-peso mexicano de los últimos 23 meses.
2. Graficar los datos obtenidos.
3. Aplicar la técnica de ajuste de mínimos cuadrados para modelar el comportamiento del tipo de cambio en función del tiempo.
4. Validar el modelo obtenido mediante la comparación de los valores predichos con los datos reales del tipo de cambio en el presente.
5. Evaluar el desempeño del modelo mediante el error cuadrático acumulado.

planteamiento:

Hicimos recolección de nuestros datos extraídos de *investing.com*, y en base a ellos se hace el siguiente planteamiento y síntesis de datos:

23

Contamos el tipo de cambio promedio de los últimos 23 meses.

error

Queremos obtener el mínimo error en nuestro modelo

grado 4

Realizaremos ajuste polinomial hasta grado 4

Graficar los datos obtenidos:

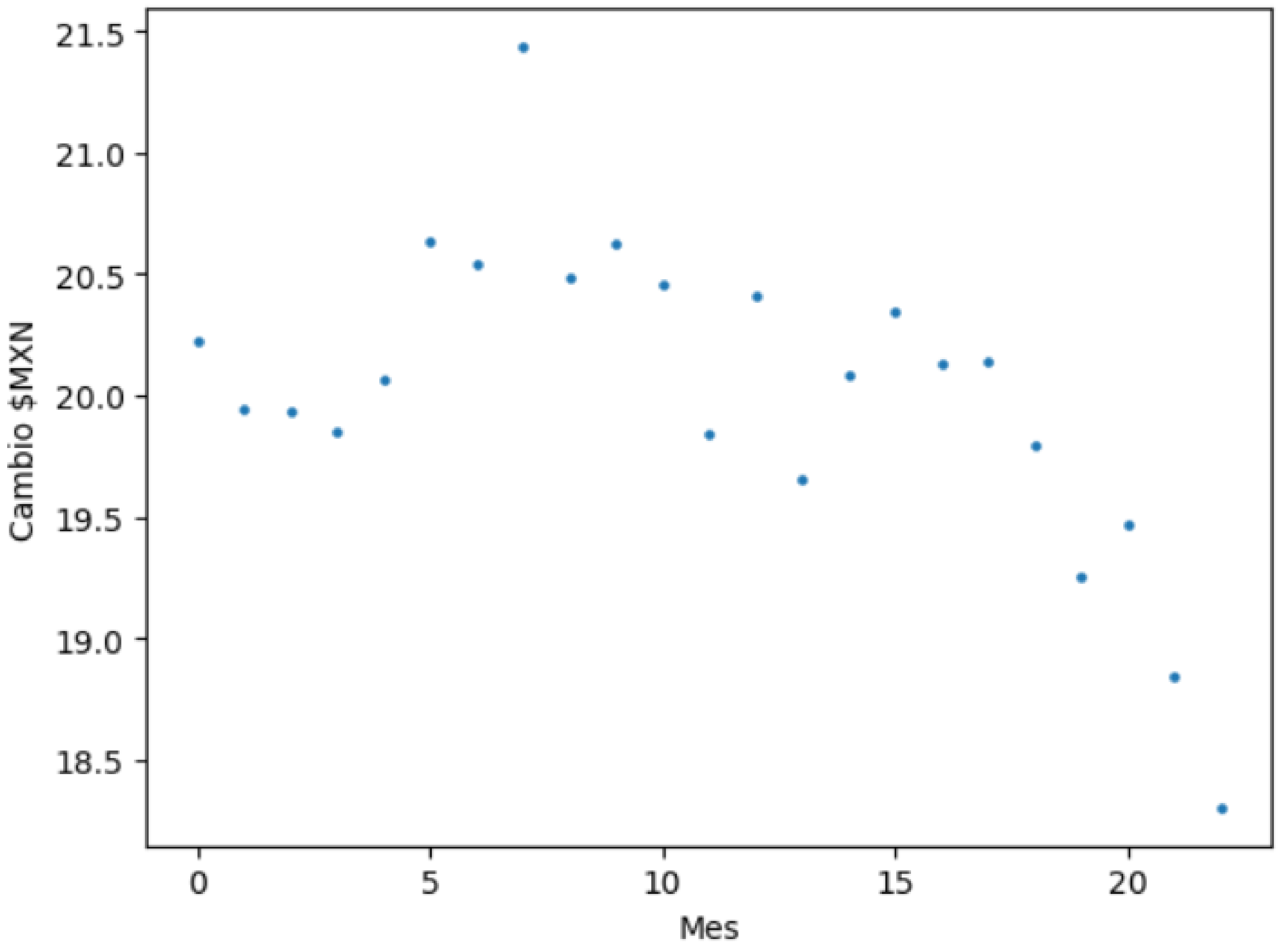


Figura 1. Gráfica del tipo de cambio USA/MXN

Los datos en "x" corresponden a los meses de 2021 a 2023, sin embargo son valores muy grandes para elevarlos al cuadrado, así que usamos la siguiente representación:

$$01.04.2021 \rightarrow 0$$
$$01.02.2023 \rightarrow 22$$

preparación de datos:

ones

np.reshape

np.hstack

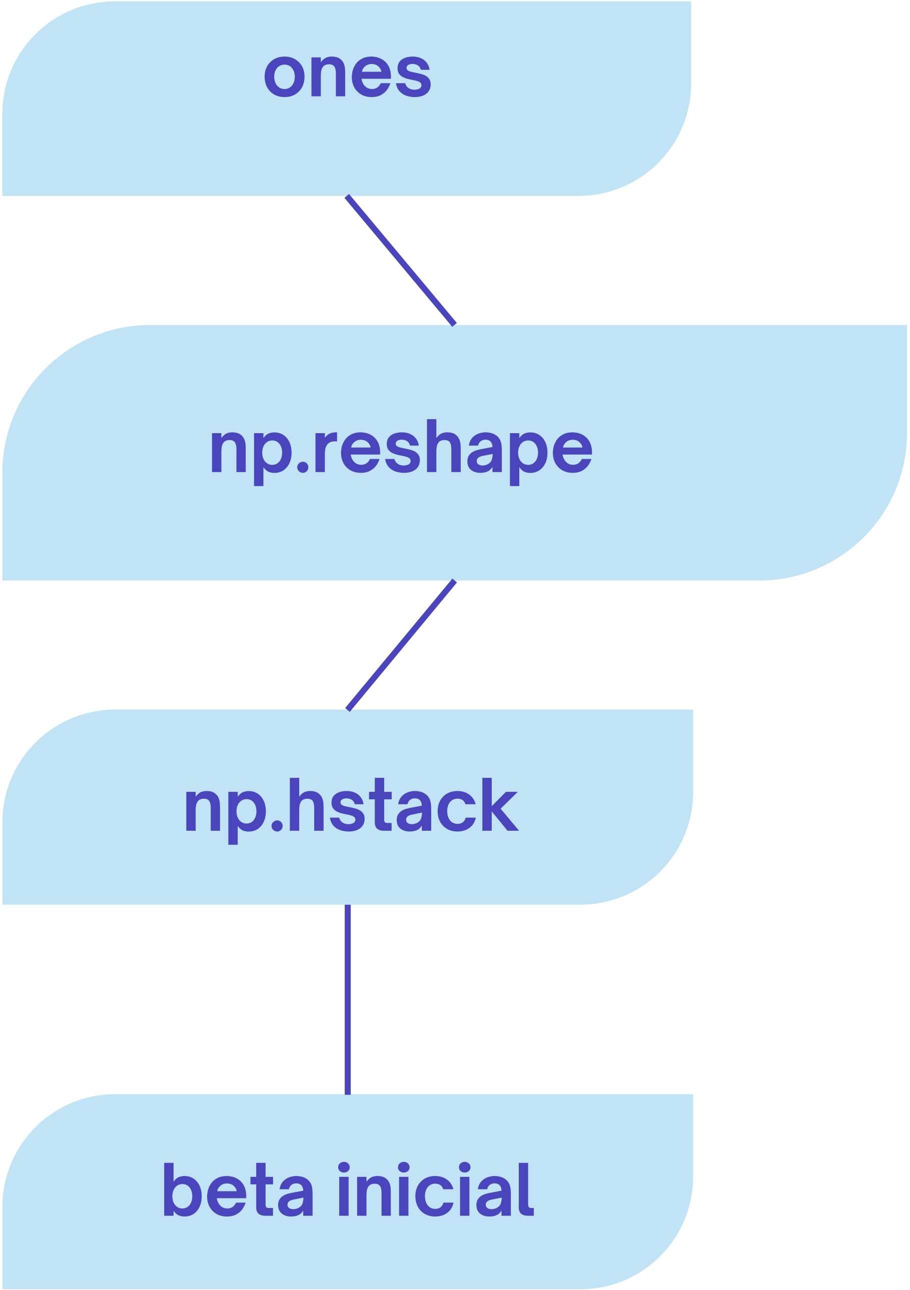
beta inicial

Creamos una matriz de 23 filas y 1 columna, donde todos los elementos de la matriz son iguales a 1.

```
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]]
```

Figura 2. Matriz "ones"

preparación de datos:

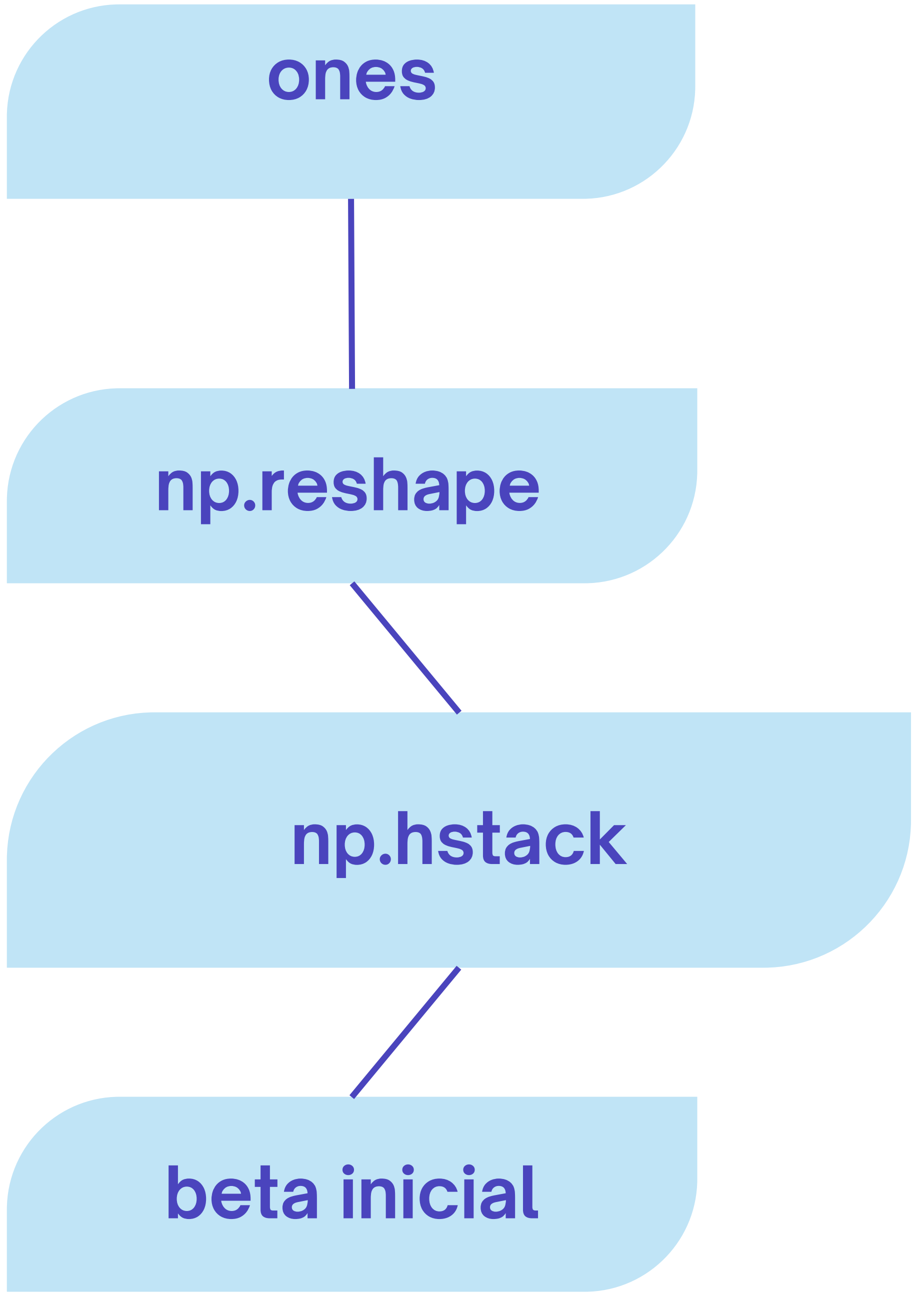


Creamos matrices de diseño polinomial para cada grado desde 1 hasta 4 utilizando la función `np.reshape` para darle forma a los array.

```
[[ 0]   [[  0]   [[    0]   [[     0]
 [ 1]    [  1]    [    1]    [     1]
 [ 2]    [  4]    [    8]    [    16]
 [ 3]    [  9]    [   27]    [    81]
 [ 4]    [ 16]    [   64]    [   256]
 [ 5]    [ 25]    [  125]    [   625]
 [ 6]    [ 36]    [  216]    [  1296]
 [ 7]    [ 49]    [  343]    [  2401]
 [ 8]    [ 64]    [  512]    [  4096]
 [ 9]    [ 81]    [  729]    [  6561]
 [10]    [100]    [ 1000]    [ 10000]
 [11]    [121]    [ 1331]    [ 14641]
 [12]    [144]    [ 1728]    [ 20736]
 [13]    [169]    [ 2197]    [ 28561]
 [14]    [196]    [ 2744]    [ 38416]
 [15]    [225]    [ 3375]    [ 50625]
 [16]    [256]    [ 4096]    [ 65536]
 [17]    [289]    [ 4913]    [ 83521]
 [18]    [324]    [ 5832]    [104976]
 [19]    [361]    [ 6859]    [130321]
 [20]    [400]    [ 8000]    [160000]
 [21]    [441]    [ 9261]    [194481]
 [22]]   [484]]   [10648]]   [234256]]
```

Figura 3. Matriz de grado 1 a grado 4

preparación de datos:



Concatenamos estas matrices de columna junto con una columna de unos, que se utiliza para ajustar el término independiente. Así creamos cuatro matrices de diseño, X1, X2, X3 y X4, que utilizaremos para estimar los coeficientes de primer a cuarto orden.

```
[[1.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00]
 [1.00000e+00 1.00000e+00 1.00000e+00 1.00000e+00 1.00000e+00]
 [1.00000e+00 2.00000e+00 4.00000e+00 8.00000e+00 1.60000e+01]
 [1.00000e+00 3.00000e+00 9.00000e+00 2.70000e+01 8.10000e+01]
 [1.00000e+00 4.00000e+00 1.60000e+01 6.40000e+01 2.56000e+02]
 [1.00000e+00 5.00000e+00 2.50000e+01 1.25000e+02 6.25000e+02]
 [1.00000e+00 6.00000e+00 3.60000e+01 2.16000e+02 1.29600e+03]
 [1.00000e+00 7.00000e+00 4.90000e+01 3.43000e+02 2.40100e+03]
 [1.00000e+00 8.00000e+00 6.40000e+01 5.12000e+02 4.09600e+03]
 [1.00000e+00 9.00000e+00 8.10000e+01 7.29000e+02 6.56100e+03]
 [1.00000e+00 1.00000e+01 1.00000e+02 1.00000e+03 1.00000e+04]
 [1.00000e+00 1.10000e+01 1.21000e+02 1.33100e+03 1.46410e+04]
 [1.00000e+00 1.20000e+01 1.44000e+02 1.72800e+03 2.07360e+04]
 [1.00000e+00 1.30000e+01 1.69000e+02 2.19700e+03 2.85610e+04]
 [1.00000e+00 1.40000e+01 1.96000e+02 2.74400e+03 3.84160e+04]
 [1.00000e+00 1.50000e+01 2.25000e+02 3.37500e+03 5.06250e+04]
 [1.00000e+00 1.60000e+01 2.56000e+02 4.09600e+03 6.55360e+04]
 [1.00000e+00 1.70000e+01 2.89000e+02 4.91300e+03 8.35210e+04]
 [1.00000e+00 1.80000e+01 3.24000e+02 5.83200e+03 1.04976e+05]
 [1.00000e+00 1.90000e+01 3.61000e+02 6.85900e+03 1.30321e+05]
 [1.00000e+00 2.00000e+01 4.00000e+02 8.00000e+03 1.60000e+05]
 [1.00000e+00 2.10000e+01 4.41000e+02 9.26100e+03 1.94481e+05]
 [1.00000e+00 2.20000e+01 4.84000e+02 1.06480e+04 2.34256e+05]]
```

Figura 4. Matriz de X4

preparación de datos:

ones

np.reshape

np.hstack

beta inicial

Iniciamos las variables de beta inicial de cada grado, con vectores con coeficientes aleatorios de longitud n + 1. Donde `n` es el grado.

```
[0.46198914 0.79537236 0.84550031 0.4589     0.21526852]
```

Figura 5. Beta inicial 4

```
[20.226  19.9461 19.931  19.855  20.063  20.63   20.545  21.4368 20.487
 20.625  20.461  19.845  20.411  19.6525 20.089  20.35   20.1348 20.14
 19.794  19.251  19.474  18.848  18.305 ]
```

Trasformamos `y` en una matriz de una columna, y 23 filas.

Matriz de "y"

Definimos una función foo que toma como argumentos un vector de parámetros beta, una matriz de diseño X, y un vector y. Esta función realiza la regresión lineal para una matriz de diseño dada utilizando el método de los mínimos cuadrados y devuelve el error cuadrático medio entre las predicciones y los valores reales de y..

Función 'foo'