# Hackatón Mty

Angela Felicia Guajardo Olveda
Brandon Alan Hernández Flores
Axel Amós Hernández Cárdenas
Izel María Ávila Rodríguez

# Metodología y variables utilizadas

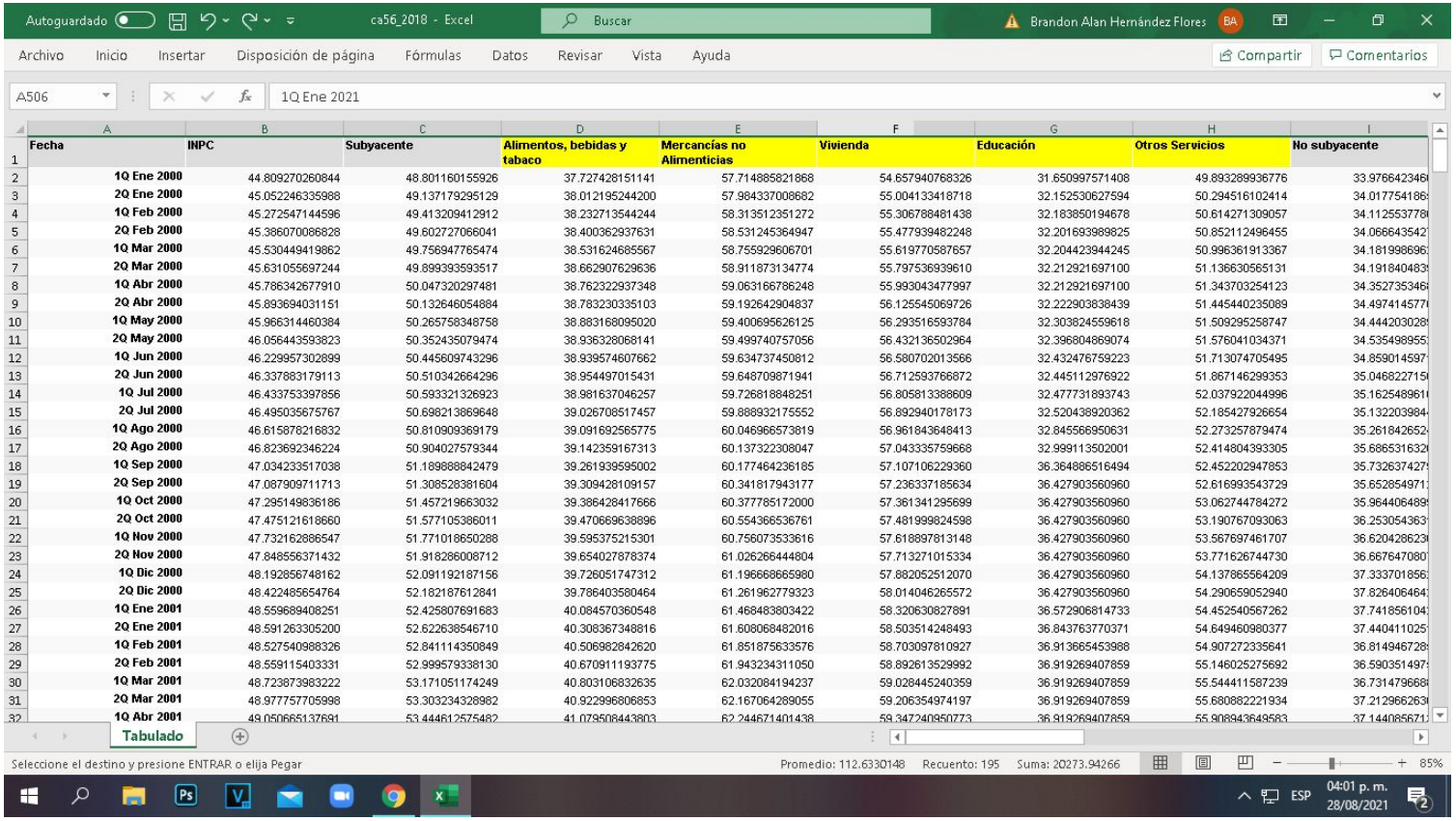
Se utilizó un programa en lenguaje Python, que es capaz de generar un modelo de predicción gracias a una base de datos con distintas variables. Después, se hace un test con otro conjunto de datos para ver sus resultados pronósticos y revisar errores. Por último, una vez valorado, se utiliza para pronosticar correctamente.

Mediante una investigación en el portal de *INEGI*, se reportaron 9 variables que influyen en el Índice Nacional de Precios al Consumidor (INPC); las primeras cinco pertenecen al conjunto de Subyacente y las siguientes cuatro pertenecen al conjunto No Subyacente.

- Alimentos, bebidas y tabaco (Mercancías)
- Mercancías no alimenticias (Mercancías)
- Vivienda (Servicios)
- Educación (colegiaturas) (Servicios)
- Otros servicios (Servicios)

- Frutas y verduras (Agropecuarios)
- Pecuarios (Agropecuarios)
- Energéticos (Energéticos y tarifas autorizadas por el gobierno)
- Tarifas autorizadas por el gobierno (Energéticos y tarifas autorizadas por el gobierno)

# Datos utilizados (Ene 2000-Ago 2021)



| | Fecha | INPC | Subyacente | Alimentos, bebidas y tabaco | Mercancías no Alimenticias | Vivienda | Educación | Otros Servicios | No subyacente |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 1Q Ene 2000 | 44.809270260844 | 48.80116055926 | 37.727428151141 | 57.714885821868 | 54.657940768326 | 31.650997571408 | 49.893289936776 | 33.976642346 |
| 3 | 2Q Ene 2000 | 45.052246335988 | 49.137179295129 | 38.012195244200 | 57.984337008682 | 55.004133418718 | 32.152530627594 | 50.294516102414 | 34.017754186 |
| 4 | 1Q Feb 2000 | 45.272547144596 | 49.413209412912 | 38.232713544244 | 58.313512351272 | 55.306788481438 | 32.183850194678 | 50.614271309057 | 34.112553778 |
| 5 | 2Q Feb 2000 | 45.386070086828 | 49.602727066041 | 38.400362937631 | 58.531245364947 | 55.477939482248 | 32.201693989825 | 50.852112496455 | 34.066643542 |
| 6 | 1Q Mar 2000 | 45.530449419862 | 49.756947765474 | 38.531624685567 | 58.755929606701 | 55.619770587657 | 32.204423944245 | 50.996361913367 | 34.181998696 |
| 7 | 2Q Mar 2000 | 45.631055697244 | 49.899393593517 | 38.662907629636 | 58.911873134774 | 55.797536939610 | 32.212921697100 | 51.136630565131 | 34.191840483 |
| 8 | 1Q Abr 2000 | 45.786342677910 | 50.047320297481 | 38.762322937348 | 59.063166786248 | 55.993043477997 | 32.212921697100 | 51.343703254123 | 34.352753466 |
| 9 | 2Q Abr 2000 | 45.893694031151 | 50.132646054884 | 38.783230335103 | 59.192642904837 | 56.125545069726 | 32.222903838439 | 51.445440235089 | 34.497414577 |
| 10 | 1Q May 2000 | 45.966314460384 | 50.265758348758 | 38.883168095020 | 59.400695626125 | 56.293516593784 | 32.303824559618 | 51.509295258747 | 34.444203028 |
| 11 | 2Q May 2000 | 46.056443593823 | 50.352435079474 | 38.936328068141 | 59.499740757056 | 56.432136502964 | 32.396804869074 | 51.576041034371 | 34.535498955 |
| 12 | 1Q Jun 2000 | 46.229957302899 | 50.445609743296 | 38.939574607662 | 59.634737450812 | 56.580702013566 | 32.432476759223 | 51.713074705495 | 34.859014597 |
| 13 | 2Q Jun 2000 | 46.337883179113 | 50.510342664296 | 38.954497015431 | 59.648709871941 | 56.712593766872 | 32.445112976922 | 51.867146299353 | 35.046822715 |
| 14 | 1Q Jul 2000 | 46.433753397856 | 50.593321326923 | 38.981637046257 | 59.726818848251 | 56.805813388609 | 32.477731893743 | 52.037922044996 | 35.162548961 |
| 15 | 2Q Jul 2000 | 46.495035675767 | 50.698213869648 | 39.026708517457 | 59.888932175552 | 56.892940178173 | 32.520438920362 | 52.185427926654 | 35.132203984 |
| 16 | 1Q Ago 2000 | 46.615878216832 | 50.810909369179 | 39.091692565775 | 60.046966573819 | 56.961843648413 | 32.845566950631 | 52.273257879474 | 35.261842652 |
| 17 | 2Q Ago 2000 | 46.823692346224 | 50.904027579344 | 39.142359167313 | 60.137322308047 | 57.043335759668 | 32.999113502001 | 52.414804393305 | 35.686531632 |
| 18 | 1Q Sep 2000 | 47.034233517038 | 51.189888842479 | 39.261939595002 | 60.177464236185 | 57.107106229360 | 36.364886516494 | 52.452202947853 | 35.732637427 |
| 19 | 2Q Sep 2000 | 47.087909711713 | 51.308528381604 | 39.309428109157 | 60.341817943177 | 57.236337185634 | 36.427903560960 | 52.616993543729 | 35.652854971 |
| 20 | 1Q Oct 2000 | 47.295149836186 | 51.457219663032 | 39.386428417666 | 60.377785172000 | 57.361341295699 | 36.427903560960 | 53.062744784272 | 35.964406489 |
| 21 | 2Q Oct 2000 | 47.475121618660 | 51.577105386011 | 39.470669638896 | 60.554366536761 | 57.481999824598 | 36.427903560960 | 53.190767093063 | 36.253054363 |
| 22 | 1Q Nov 2000 | 47.732162886547 | 51.771018650288 | 39.595375215301 | 60.756073533616 | 57.618897813148 | 36.427903560960 | 53.567697461707 | 36.620428623 |
| 23 | 2Q Nov 2000 | 47.848556371432 | 51.918286008712 | 39.654027878374 | 61.026266444804 | 57.713271015334 | 36.427903560960 | 53.771626744730 | 36.667647080 |
| 24 | 1Q Dic 2000 | 48.192856748162 | 52.091192187156 | 39.726051747312 | 61.196686865980 | 57.882052512070 | 36.427903560960 | 54.137865564209 | 37.333701856 |
| 25 | 2Q Dic 2000 | 48.422485654764 | 52.182187612841 | 39.786403580464 | 61.261962779323 | 58.014046265572 | 36.427903560960 | 54.290659052940 | 37.826406464 |
| 26 | 1Q Ene 2001 | 48.559689408251 | 52.425807691683 | 40.084570360548 | 61.468483803422 | 58.320630827891 | 36.572906814733 | 54.452540567262 | 37.741856104 |
| 27 | 2Q Ene 2001 | 48.591263305200 | 52.622638546710 | 40.308367348816 | 61.608068482016 | 58.503514248493 | 36.843763770371 | 54.649460980377 | 37.440411025 |
| 28 | 1Q Feb 2001 | 48.527540988326 | 52.841114350849 | 40.506982842620 | 61.851875633576 | 58.703097810927 | 36.913665453988 | 54.907272335641 | 36.814946728 |
| 29 | 2Q Feb 2001 | 48.559115403331 | 52.999579338130 | 40.670911193775 | 61.943234311050 | 58.892613529992 | 36.919269407859 | 55.146025275692 | 36.590351497 |
| 30 | 1Q Mar 2001 | 48.723873983222 | 53.171051174249 | 40.803106832635 | 62.032084194237 | 59.028445240359 | 36.919269407859 | 55.544411587239 | 36.731479668 |
| 31 | 2Q Mar 2001 | 48.977757705998 | 53.303234328982 | 40.922996806853 | 62.167064289055 | 59.206354974197 | 36.919269407859 | 55.680882221934 | 37.212966263 |
| 32 | 1Q Abr 2001 | 49.050665137691 | 53.444612575482 | 41.079508443803 | 62.244671401436 | 59.347240950773 | 36.919269407859 | 55.908943649583 | 37.144085671 |

Información obtenida del Portal de INEGI: INEGI:https://www.inegi.org.mx/temas/inpc/

# Coeficientes obtenidos

| | Coeficientes |
|---|---|
| Alimentos, bebidas y tabaco | 0.214784 |
| Mercancías no Alimenticias | 0.145089 |
| Vivienda | 0.197848 |
| Educación | 0.047456 |
| Otros Servicios | 0.133276 |
| Frutas y Verduras | 0.040857 |
| Pecuarios | 0.050407 |
| Energéticos | 0.103877 |
| Tarifas autorizadas por el gobierno | 0.059408 |

Modelo de regresión lineal múltiple

INPC = 0.214784(Alimentos bebidas y tabaco)+ 0.145089(Mercancías no Alimenticias)+ 0.197848(Vivienda)+ 0.047456(Educación)+ 0.133276(Otros Servicios)+ 0.040857 (Frutas y Verduras)+ 0.050407(Pecuarios)+ 0.103877(Energéticos)+ 0.059408(Tarifas autorizadas por el gobierno)

# Parte 1 - Script (ENTRENAMIENTO DE PROGRAMA Y DIVISIÓN DE MUESTRA 1Q ENE-2000, 2Q DIC-2020)

PANDAS, Lectura, Descripción de datos

```
import pandas as pd # Librería pandas
datos_INPC = pd.read_excel( 'ca56_2018.xlsx' ) # indicamos el nombre de nuestro archivo a ser leído
datos_INPC.head()  #Se pone el encabezado para ver cuáles fueron los atributos cargados.
datos_INPC.describe()
```

Selección, Mostrar, Valores Nulos

```
datos_seleccionados = datos_INPC.iloc[:, 1:15] # : selecciona todas las filas y :,1:16 seleccion columnas de la 1 la 16
datos_seleccionados  # desplegamos el dataframe
datos_seleccionados.info()
datos_seleccionados.isnull().values. any() # buscamos valores nulos y obtenemos True o False dependiendo si hay o no
```

Eliminar Nulos, validación

```
dataset = datos_seleccionados.dropna()  # creamos un nuevo dataframe descartando los valores nulos o vacíos de nuestro dataframe datos_seleccionados
dataset.isnull(). sum() # validamos que no tenemos valores nulos en ninguna columna, todos deben dar cero
```

Nombrar Colum., 'X y Y' data

```
dataset.columns  # vemos los nombres de nuestras columnas para asignarlos a las variables
X = dataset[['Alimentos, bebidas y tabaco' , 'Mercancías no Alimenticias' , 'Vivienda', 'Educación', 'Otros Servicios' , 'Frutas y Verduras' , 'Pecuarios', 'Energéticos']].values # variables independientes
y = dataset['INPC'].values # variable dependiente
```

# Parte 2 - Script

80% entren.
20% test

```
from sklearn.model_selection import train_test_split # importamos la herramienta para dividir los datos de SciKit-Learn
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0) # asignación de los datos 80% para entrenamiento y 20% para prueba
```

Modelo de regresión, despliegue de Coef.

```
from sklearn.linear_model import LinearRegression # importamos la clase de regresión lineal
modelo_regresion = LinearRegression() # modelo de regresión
modelo_regresion.fit(X_train, y_train) # aprendizaje automático con base en nuestros datos
x_columns = ['Alimentos, bebidas y tabaco', 'Mercancías no Alimenticias', 'Vivienda', 'Educación', 'Otros Servicios', 'Frutas y Verduras', 'Pecuarios', 'Energéticos']# Aquí se estan seleccionando los datos base, que se mencionaba arriba.
coeff_df = pd.DataFrame(modelo_regresion.coef_, x_columns, columns=['Coeficientes'])
coeff_df # despliega los coeficientes y sus valores; por cada unidad del coeficiente, su impacto en las calorías será igual a su valor
```

# Parte 3 - Script

Test, errores

Error Medio, Std, Max, min

Coef. Determinación

Matplotlib.pyplot, graficar

```
y_pred = modelo_regresion.predict(X_test)  # probamos nuestro modelo con los valores de prueba
validacion = pd.DataFrame({ 'Actual': y_test, 'Predicción': y_pred, 'Diferencia': y_test-y_pred })  # creamos un dataframe con los valores actuales y los de predicción
muestra_validacion = validacion.head( 100) # elegimos una muestra con 100 valores
muestra_validacion  # desplegamos esos 100 valores
validacion["Diferencia"].describe()
from sklearn.metrics  import  r2_score # importamos la métrica R cuadrada (coeficiente de determinación)
r2_score(y_test, y_pred)  # ingresamos nuestros valores reales y calculados para  obtener la R2
import matplotlib.pyplot as plt # importamos la librería que nos permitirá graficar
muestra_validacion.plot.bar(rot=0)  # creamos un gráfico de barras con el dataframe que contiene nuestros datos actuales y de predicción
plt.title("INPC 1Q ENE-2000, 2Q DIC-2020") # indicamos el título del gráfico
plt.xlabel("Muestra quincenal") # indicamos la etiqueta del eje de las x
plt.ylabel("Valor INPC") # indicamos la etiqueta del eje de las y
plt.show() # desplegamos el gráfico
```

# Parte 4 - Script (PREDICCIÓN INPC 1Q ENE-2021, 1Q AGO-2021)

Lectura

Selección, Mostrar, Valores Nulos

Eliminar Nulos, validación

Nombrar Colum., 'X y Y' data

```
datos_INPC2 = pd.read_excel( 'Pronósticos.xlsx' ) # indicamos el nombre de nuestro archivo a ser leído
datos_seleccionados2 = datos_INPC2.iloc[:, 1:15] # : selecciona todas las filas y columnas
datos_seleccionados2  # desplegamos el dataframe
datos_seleccionados2.info()
datos_seleccionados2.isnull().values. any() # buscamos valores nulos y obtenemos True o False
dependiendo si hay o no
dataset2 = datos_seleccionados2.dropna()  # creamos un nuevo dataframe descartando los valores nulos
o vacíos de nuestro dataframe datos_seleccionados
dataset2.isnull(). sum() # validamos que no tenemos valores nulos en ninguna columna, todos deben dar
cero
dataset2.columns  # vemos los nombres de nuestras columnas para asignarlos a las variables
X2 = dataset2[['Alimentos, bebidas y tabaco' , 'Mercancías no Alimenticias' , 'Vivienda', 'Educación',
'Otros Servicios', 'Frutas y Verduras' , 'Pecuarios', 'Energéticos']].values # variables
independientes
y2 = dataset2['INPC'].values # variable dependiente
y_pred2 = modelo_regresion.predict(X2)
```

# Parte 5

Errores, mostrar validación

Coef. Determinación

Matplotlib.pyplot, graficar

```
validacion2 = pd.DataFrame({'Actual': y2, 'Predicción': y_pred2, 'Diferencia': y2-y_pred2 })
# creamos un dataframe con los valores actuales y los de predicción
muestra_validacion2 = validacion2.head(15) # elegimos una muestra con 15 valores
muestra_validacion2 # desplegamos esos 15 valores
from sklearn.metrics import r2_score # importamos la métrica R cuadrada (coeficiente de determinación)
r2_score(y2, y_pred2) # ingresamos nuestros valores reales y calculados para obterner la R2
import matplotlib.pyplot as plt # importamos la librería que nos permitirá graficar
muestra_validacion2.plot.bar(rot=0)  # creamos un gráfico de barras con el dataframe que contiene nuestros datos actuales y de predicción
plt.title("INPC 1Q ENE-2021, 1Q AGO-2021") # indicamos el título del gráfico
plt.xlabel("Muestras quincenales") # indicamos la etiqueta del eje de las x
plt.ylabel("Valor INPC") # indicamos la etiqueta del eje de las y
plt.show() # desplegamos el gráfico
```

# Parte 6 (PRONÓSTICOS INPC SUBYACENTE 1Q ENE-2021, 1Q AGO-2021)

‘X4 y Y4’ data

Pronóstico, errores

Promedio, std, min, max de error

Matplotlib.pyplot, graficar

```
X4 = dataset2[['Alimentos, bebidas y tabaco', 'Mercancías no Alimenticias', 'Vivienda',
'Educación', 'Otros Servicios', 'Control', 'Control', 'Control','Control']].values # variables
independientes
y4 = dataset2['Subyacente'].values # variable dependiente
y_pred4 = modelo_regresion.predict(X4)
validacion4 = pd.DataFrame({'Actual': y4, 'Predicción': y_pred4, 'Diferencia': y4-y_pred4}) #
creamos un dataframe con los valores actuales y los de predicción
muestra_validacion4 = validacion4.head(15) # elegimos una muestra con 25 valores
muestra_validacion4 # desplegamos esos 25 valores
validacion4["Diferencia"].describe()
import matplotlib.pyplot as plt # importamos la librería que nos permitirá graficar
muestra_validacion4.plot.bar(rot=0)  # creamos un gráfico de barras con el dataframe que contiene
nuestros datos actuales y de predicción
plt.title("SUBYACENTE 1Q ENE-2021, 1Q AGO-2021") # indicamos el título del gráfico
plt.xlabel("Muestras quincenales") # indicamos la etiqueta del eje de las x, los alimentos
plt.ylabel("Valor Subyacente") # indicamos la etiqueta del eje de las y, la cantidad de calorías
plt.show() # desplegamos el gráfico
```

# Parte 7 (PRONÓSTICOS VARIACIÓN ANUAL INPC SUBYACENTE DENTRO DE LA MUESTRA)

Lectura

Selección, Mostrar, Valores Nulos

Eliminar Nulos, validación

Nombrar Colum., 'X y Y' data

80% y 20%

```
import pandas as pd # Librería pandas
datos_INPC3 = pd.read_excel('ca56_2018.xlsx') # indicamos el nombre de nuestro archivo a ser leído
datos_seleccionados3 = datos_INPC3.iloc[:,2:15]
datos_seleccionados3
datos_seleccionados3.info()
datos_seleccionados3.isnull().values.any() # buscamos valores nulos y obtenemos True o False dependiendo si hay o no
dataset3 = datos_seleccionados3.dropna() # creamos un nuevo dataframe descartando los valores nulos o vacíos de nuestro dataframe datos_seleccionados
dataset3.isnull().sum() # validamos que no tenemos valores nulos en ninguna columna, todos deben dar cero
dataset3.columns # vemos los nombres de nuestras columnas para asignarlos a las variables
X3 = dataset3[['Alimentos, bebidas y tabaco', 'Mercancías no Alimenticias', 'Vivienda', 'Educación', 'Otros Servicios','Control','Control','Control','Control']].values # variables independientes
y3 = dataset3['Subyacente'].values # variable dependiente
from sklearn.model_selection import train_test_split # importamos la herramienta para dividir los datos de SciKit-Learn
X_train3, X_test3, y_train3, y_test3 = train_test_split(X3, y3, test_size=0.2, random_state=0) # asignación de los datos 80% para entrenamiento y 20% para prueba
```

# Parte 8

Errores, mostrar validación

```
y_pred3 = modelo_regresion.predict(X_test3)
validacion3 = pd.DataFrame({'Actual': y_test3, 'Predicción': y_pred3, 'Diferencia':
y_test3-y_pred3 }) # creamos un dataframe con los valores actuales y los de predicción
muestra_validacion3 = validacion3.head(100) # elegimos una muestra con 25 valores
muestra_validacion3 # desplegamos esos 25 valores
validacion3["Diferencia"].describe()
```

Matplotlib.pyplot, graficar

```
import matplotlib.pyplot as plt # importamos la librería que nos permitirá graficar
muestra_validacion3.plot.bar(rot=0)  # creamos un gráfico de barras con el dataframe que
contiene nuestros datos actuales y de predicción
plt.title("SUBYACENTE 1Q ENE-2000, 2Q DIC-2020") # indicamos el título del gráfico
plt.xlabel("Muestras quincenales") # indicamos la etiqueta del eje de las x, los alimentos
plt.ylabel("Valor Subyacente") # indicamos la etiqueta del eje de las y, la cantidad de
calorías
plt.show() # desplegamos el gráfico
```

# Resultados

INPC 1Q ENE-2000, 2Q DIC-2020

| | Actual | Predicción | Diferencia |
|---|---|---|---|
| 0 | 54.701944 | 54.723555 | -0.021612 |
| 1 | 55.863423 | 55.860332 | 0.003091 |
| 2 | 105.299000 | 105.339403 | -0.040403 |
| 3 | 84.943506 | 84.962522 | -0.019016 |
| 4 | 89.327784 | 89.362333 | -0.034549 |
| ... | ... | ... | ... |
| 95 | 94.852455 | 94.822614 | 0.029841 |
| 96 | 51.483587 | 51.578522 | -0.094936 |
| 97 | 99.818949 | 99.819141 | -0.000192 |
| 98 | 51.872845 | 51.942207 | -0.069362 |
| 99 | 107.777000 | 107.741927 | 0.035073 |

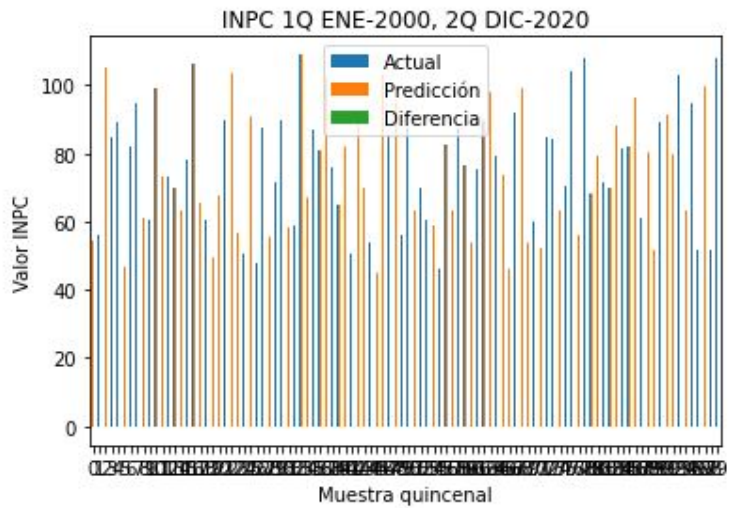
```
validacion["Diferencia"].describe(
```

```
count    101.000000
mean      -0.002725
std        0.037445
min       -0.094936
25%       -0.028335
50%       -0.001640
75%        0.025176
max        0.078848
Name: Diferencia, dtype: float64
```

```
r2_score(y_test, y_pred) #
```

```
0.9999836651941661
```

```
plt.show() # desplegamos el gráfico
```



# Resultados

## PREDICCIÓN INPC 1Q ENE-2021, 1Q AGO-2021

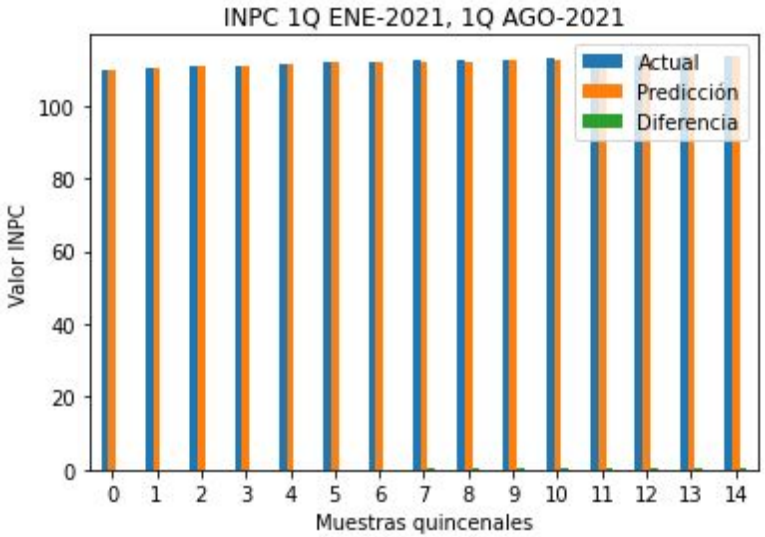
| | Actual | Predicción | Diferencia |
|---|---|---|---|
| 0 | 109.936 | 109.893619 | 0.042381 |
| 1 | 110.484 | 110.445251 | 0.038749 |
| 2 | 110.736 | 110.696511 | 0.039489 |
| 3 | 111.079 | 111.024035 | 0.054965 |
| 4 | 111.668 | 111.583662 | 0.084338 |
| 5 | 111.981 | 111.881208 | 0.099792 |
| 6 | 112.052 | 111.912121 | 0.139879 |
| 7 | 112.328 | 112.162964 | 0.165036 |
| 8 | 112.321 | 112.099987 | 0.221013 |
| 9 | 112.517 | 112.294374 | 0.222626 |
| 10 | 112.903 | 112.647600 | 0.255400 |
| 11 | 113.132 | 112.873866 | 0.258134 |
| 12 | 113.547 | 113.273938 | 0.273062 |
| 13 | 113.818 | 113.542047 | 0.275953 |
| 14 | 113.794 | 113.476160 | 0.317840 |

```
validacion2["Diferencia"].describe()
```

```
count    15.000000
mean      0.165911
std       0.100308
min       0.038749
25%       0.069652
50%       0.165036
75%       0.256767
max       0.317840
Name: Diferencia, dtype: float64
```

```
r2_score(y2, y_pred2)
```

```
0.9726641825809438
```



# Resultados

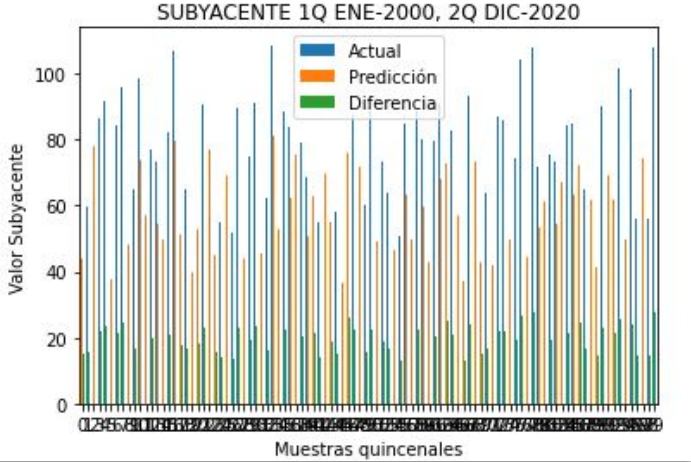
SUBYACENTE 1Q ENE-2000, 2Q DIC-2020

| | Actual | Predicción | Diferencia |
|---|---|---|---|
| 0 | 59.055924 | 43.769014 | 15.286910 |
| 1 | 59.701246 | 44.278880 | 15.422366 |
| 2 | 104.798292 | 78.146390 | 26.651902 |
| 3 | 86.521993 | 64.494718 | 22.027275 |
| 4 | 91.808920 | 68.457137 | 23.351783 |
| ... | ... | ... | ... |
| 95 | 95.363688 | 71.086981 | 24.276708 |
| 96 | 56.097507 | 41.566225 | 14.531282 |
| 97 | 99.948584 | 74.515627 | 25.432957 |
| 98 | 56.246024 | 41.709364 | 14.536660 |
| 99 | 108.034618 | 80.568590 | 27.466028 |

```
validacion3["Diferencia"].describe()
```

```
count    101.000000
mean      19.908662
std        4.078793
min       12.733623
25%       16.472501
50%       19.913177
75%       23.033543
max       27.620178
Name: Diferencia, dtype: float64
```

```
plt.show() # desplegamos el gráfico
```



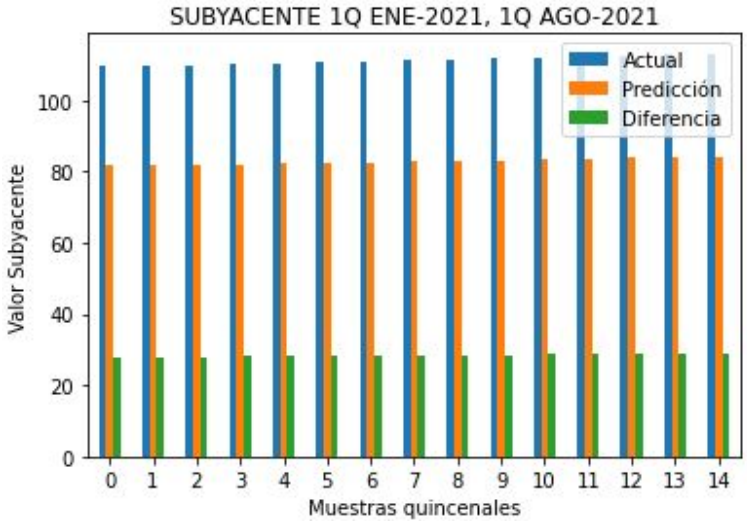
# Resultados

PREDICCIÓN SUBYACENTE 1Q ENE-2021, 1Q AGO-2021

| | Actual | Predicción | Diferencia |
|---|---|---|---|
| 0 | 109.535926 | 81.637823 | 27.898103 |
| 1 | 109.721867 | 81.770602 | 27.951266 |
| 2 | 109.961759 | 81.934137 | 28.027622 |
| 3 | 110.160047 | 82.067594 | 28.092452 |
| 4 | 110.541054 | 82.324682 | 28.216372 |
| 5 | 110.766903 | 82.483664 | 28.283239 |
| 6 | 110.966931 | 82.623787 | 28.343144 |
| 7 | 111.154931 | 82.758986 | 28.395945 |
| 8 | 111.520667 | 83.012290 | 28.508377 |
| 9 | 111.767779 | 83.196180 | 28.571598 |
| 10 | 112.163713 | 83.459706 | 28.704007 |
| 11 | 112.398553 | 83.628147 | 28.770406 |
| 12 | 112.742999 | 83.872906 | 28.870093 |
| 13 | 112.887154 | 83.989397 | 28.897757 |
| 14 | 113.202701 | 84.217823 | 28.984879 |

```
validacion4["Diferencia"].describe()
```

```
count    15.000000
mean     28.434351
std       0.358674
min      27.898103
25%      28.154412
50%      28.395945
75%      28.737207
max      28.984879
Name: Diferencia, dtype: float64
```



# Referencias

INEGI. (2021, Julio). *Índice Nacional de Precios de Consumidor*. INEGI:https://www.inegi.org.mx/temas/inpc/