Trabajo Final de Evaluación

Universidad del Desarrollo

Universidad de Excelencia

Curso Python para Análisis de Datos

Trabajo	IPC – Especial Alimentos
Integrantes	Alfaro, Christian
	Briceño, Heriberto
	Rovai, Marcelo
	Sacasa, Manuel

Santiago, 25 de agosto de 2018

1. Introducción

Este trabajo es parte del proceso de evaluación conjunta de cursos: Python para Análisis de Datos y Macroeconomía para Data Science (Versión 2018). Los cursos son impartidos por los profesores Ismael Botti y Mercedes Haga para el Magister/Diplomado en Data Science de la Escuela de Ingeniería de la Universidad del Desarrollo – UDD – Campus de Santiago, Chile.

2. Observaciones generales acerca del trabajo:

2.1. Descarga de la información desde el sitio web ODEPA:

El script ***ipc_especial.py*** debe de estar en la misma carpeta donde se encuentra la carpeta principal de datos ("data/"). Esta carpeta de datos deberá contener subcarpetas donde estarán las bases de datos regionales. Por ejemplo:

```
|__ ipc_especial.py
|__ data
      |__ data_ar        (Data - Arica)
      |__ data_rm        (Data - Metropolitana consolidada)
      |__ data_po        (Data - Metropolitana poniente)
      |__ data_no        (Data - Metropolitana norte)
      |__ …              ...
```

Los datasets deberán ser descargados del sitio de la ODEPA. Abajo, un ejemplo de las opciones para el caso de Arica :

- **Tipo de Serie:** Mensual
- **Periodo:** Desde 06/2014 hasta 07/2018
- **Regiones:** Arica y Metropolitana (Todos los sectores)
- **Tipos de Productos:** Todos (excepto Aceitunas)
- **Tipo punto monitoreo:** Supermercado
- **Pesos:** Nominales

Note que el site de ODEPA retornará el informe partindo del mes más antiguo disponíble desde sus bases. Por ejemplo, al introducir “Desde 06/2014”…, el informe retornará con la fecha de 10/2014.

La Ilustración 1 muestra un ejemplo de la interfaz para descarga del informe en el sitio de ODEPA.

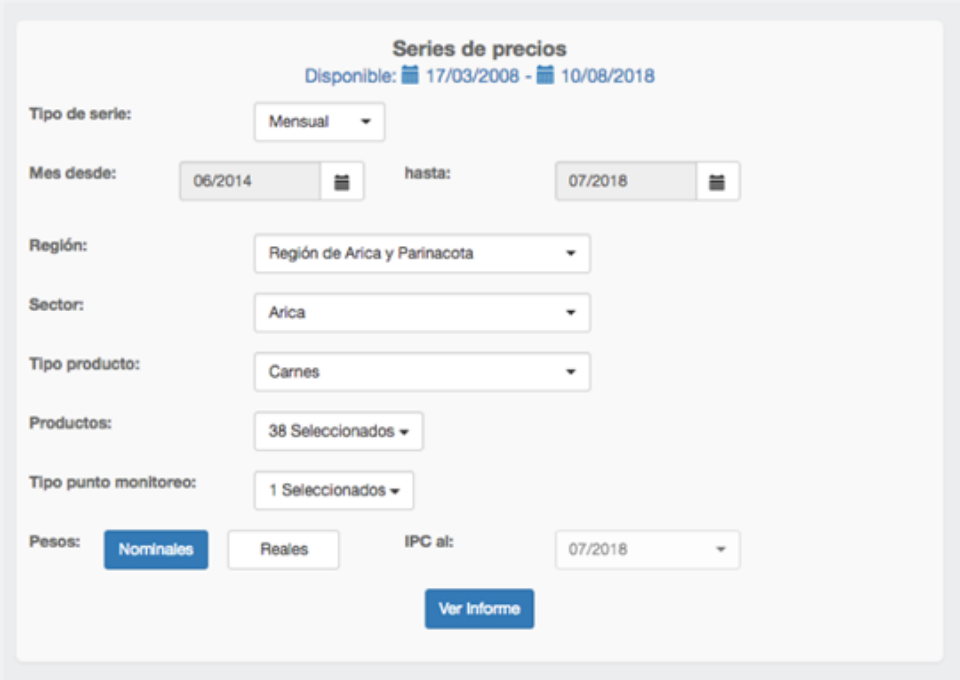

Ilustración 1 - Interfaz para descarga del Informe Tipo de prodcto: "Carnes" para la región de Arica y Parinacota

Cada una de las subcarpetas, deberá contener los 11 ‘informes’ bajados del sitio de ODEPA. Importante cambiar el nombre original de cada informe bajado del sitio, en acuerdo a la siguiente nomenclatura:

/data/**data_yy**/**Producto**_SeriesDePrecios.xlsx

Descripción:

- Producto es Aceite, Pan, etc.
- data_yy es data_ar (Arica), data_rm (Reg. Metropolitana consolidada), etc.

La ilustración 2 muestra como quedaría la carpeta para Arica.

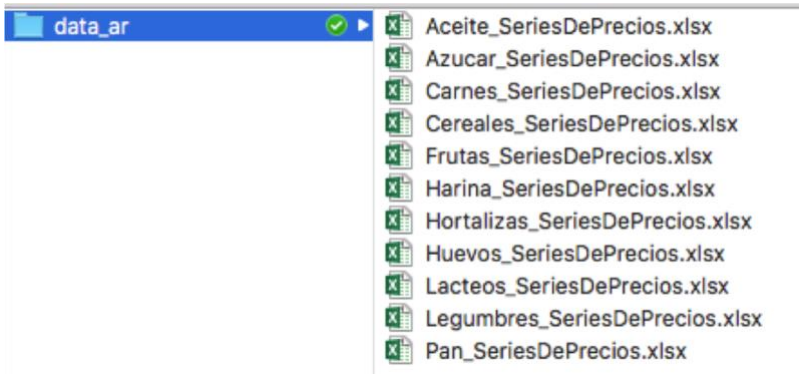

Ilustración 2 - Carpeta de datos: Arica

2.2 Creación de una tabla consolidada para los IPCs regionales:

- Importar el módulo ***ipc_especial.py***, con las funciones que serán utilizadas, a través de uno de los siguientes métodos:
    - *import ipc_especial as ipc* => ej: llamada: *ipc.monta_ipc(data)*
    - *from ipc_especial import ** => ej: llamada: *monta_ipc(data)*

- Utilizar la función *monta_ipc(region, graph=True)*, que a partir de la carpeta donde se encuentran los datos de una región específica (por ejemplo, Arica: /data_ar), retornará una tabla completa conteniendo el índice del IPC de la región y los componentes por producto, teniendo la fecha de el mes 2 (en el caso de este trabajo: 11/2014) del informe descargado del sitio de ODEPA como mes base (100). La función también retorna como default una gráfica con la variación del IPC regional. Al llamar la función con el parámetro *graph=False*, la gráfica no será ejecutada.

- Todas las regiones deberán ser bajadas con la función *monta_ipc()*. Por ejemplo, para Arica y Metropolitana consolidados:

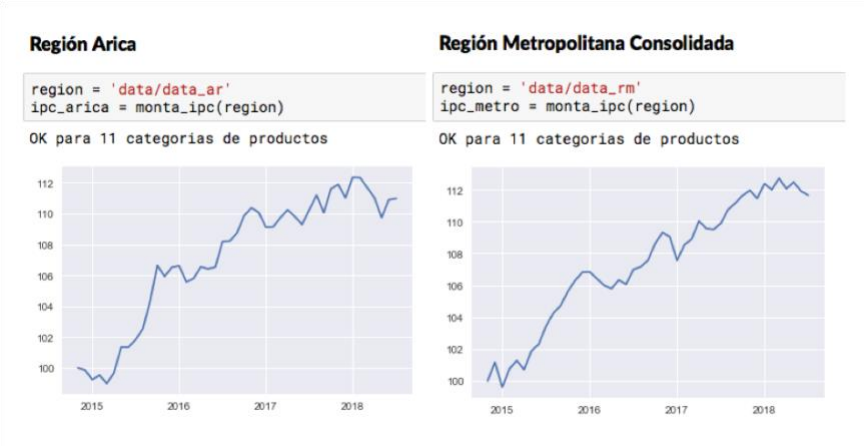

Ilustración 3 - Utilización de la función monta_ipc() para carga del dataset

- Una vez cargadas las regiones, las mismas podrán ser concatenadas, en una única tabla teniendo como columnas los IPCs de cada Región. Entonces se podrá hacer análisis de los datos. Por ejemplo, para las dos regiones ya cargadas:

```
ipc = pd.DataFrame(ipc_arica['IPC-Indice'])
```

```
ipc = pd.concat([ipc, ipc_metro['IPC-Indice']], axis =1)
ipc.columns = ['Arica', 'Metropolitana']
ipc.head()
```

	Arica	Metropolitana
2014-11-01	100.000000	100.000000
2014-12-01	99.849657	101.176324
2015-01-01	99.240102	99.605269
2015-02-01	99.531996	100.753942
2015-03-01	98.989524	101.274320

```
ipc.plot(figsize=(10,5), ylim = (98,120))
<matplotlib.axes._subplots.AxesSubplot at 0x1a21769588>
```

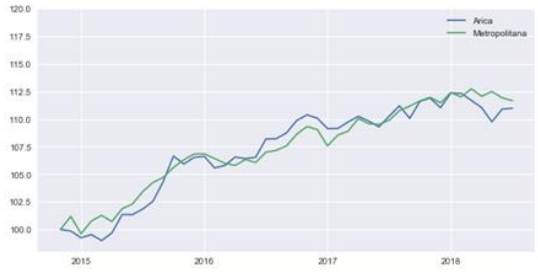

Ilustración 4 - Datasets consolidados

3. Ejemplos de Análisis del IPC-Especial ODEPA

3.1. IPC-Especial ODEPA – Correlación con indice del INE:

Se podrá bajar otros datasets y compararlos con el dataset creado a partir de los datos de la ODEPA. Por ejemplo, se puede comparar el dataset con los IPCs totales para Chile (División Alimentos y Bebidas no Alcohólicas), bajado del sitio del INE , tomándose como base 100, Noviembre/14.

Luego de esto es Interesante verificar que existe una correlación entre el IPC-Especial obtenido en este trabajo y el IPC Real a pesar de que solamente fueron considerados los supermercados como fuente. Además la canasta entre los dos índices no son exactamente iguales.

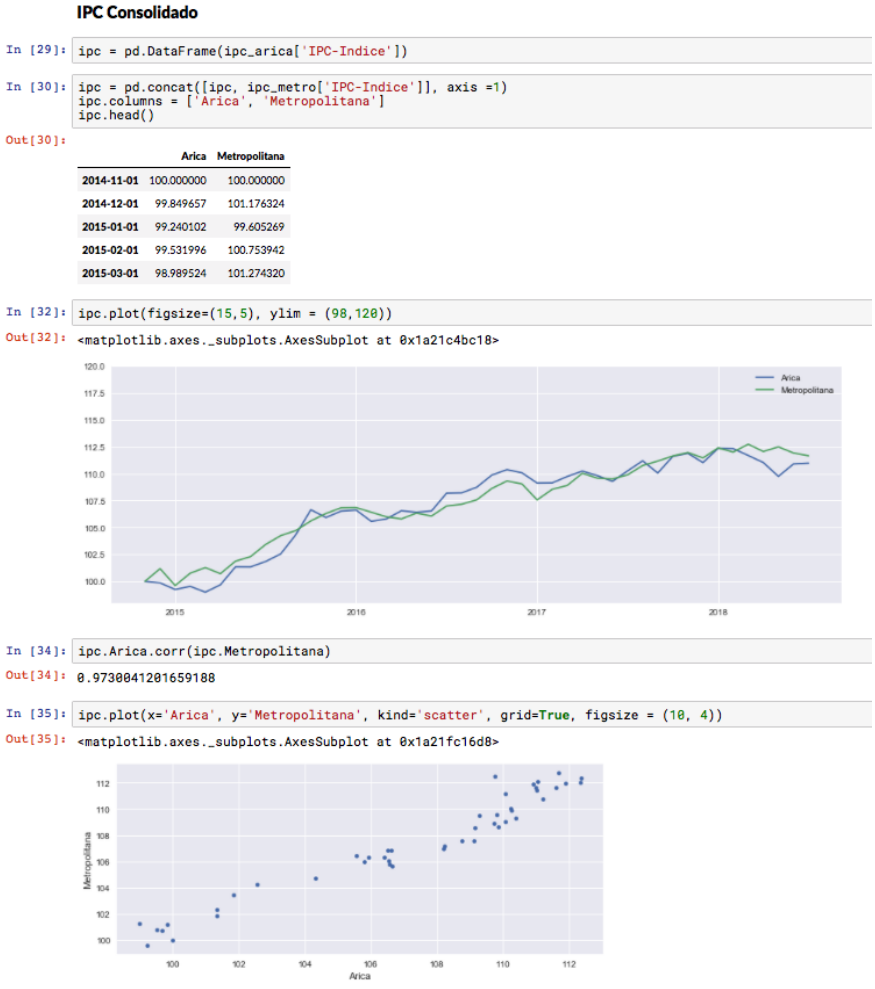

Ilustración 5 - IPC Consolidado y su corrrelación con IPC Alimentos Chile

Otros índices interesantes para se comparar con el IPC-Alimentos són “Dólar” y “IPC-Diesel”.

3.2. Machine Learning:

Otro análisis interesante es aplicar técnicas de Machine Learning (regresión lineal) a los datos y mirar cómo se comportan los índices entre los datos consolidados de las regiones de Arica y Metropolitana. Se podrá observar que los precios de alimentos en la Región Metropolitana van a ser algo como 0.5% más bajos que la Región de Arica, para un indice futuro de 115:

```
from sklearn.linear_model import LinearRegression
```

```
def ml_reg_lin(df, reg1, reg2, x_in=100):

    x = df[reg1]
    y = df[reg2]
    model = LinearRegression(fit_intercept=True)
    X = x[:, np.newaxis]
    model.fit(X, y)
    xfit = np.linspace(98, 115)
    Xfit = xfit[:, np.newaxis]
    yfit = model.predict(Xfit)

    y_out = model.predict(x_in)[0]
    print("\n[Predición] Indice de {} en la Región {}, equivale a {} en la Region {}\n"
          .format(x_in, reg1, round(y_out, 2), reg2))

    plt.scatter(x, y)
    plt.plot(xfit, yfit)
```

```
reg1 = 'Arica'
reg2 = 'Metropolitana'

ml_reg_lin(ipc_test, reg1, reg2, 115)
```

```
[Predición] Indice de 115 en la Región Arica, equivale a 114.43 en la Region Metropolitana
```

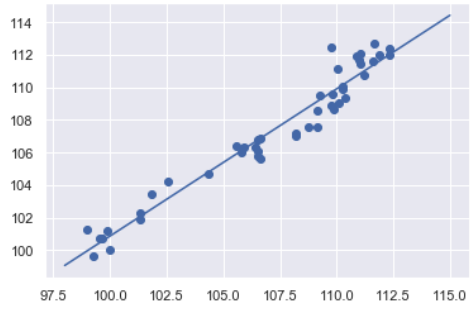

Ilustración 6 - Aplicando Machine Learning

4. Ejecutando el código desde una terminal

- Para ejecutar el programa en la terminal, se debe ingresar el siguiente comando:

  Python ipc_especial.py

- El resultado de la ejecución del script informará con “OK” para cada carga de región y las primeras 5 primeras líneas del dataset consolidado será presentada, además de la gráfica con las series de IPC temporales.

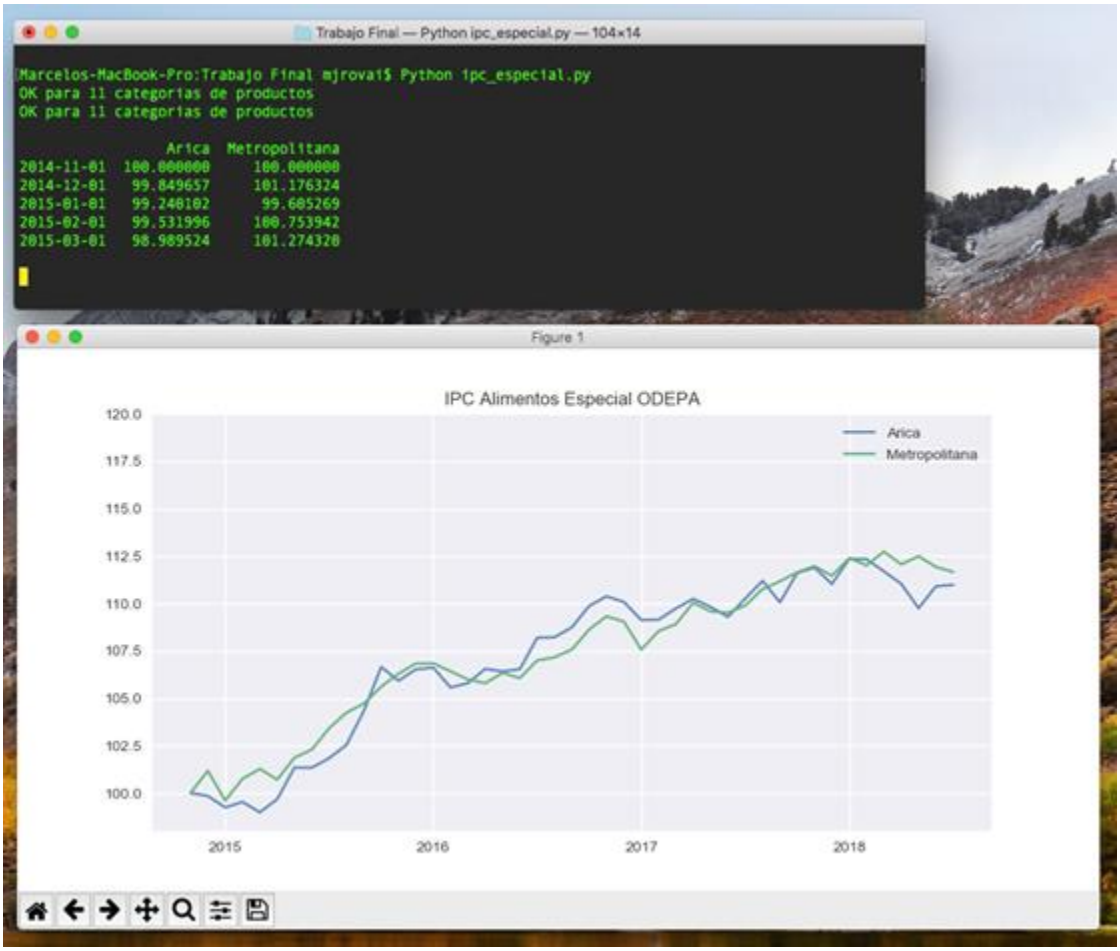

Ilustración 7 - Ejecución del programa en la terminal

5. Descripción de funciones presentes en el código:

El script desarrollado para el módulo ***ipc_especial.py***, está compuesto de 10 funciones principales, cuyas explicaciones están a continuación:

La función ***monta_ipc()***, utilizada en el trabajo, internamente llama 3 funciones:

- *carga_region()* => **pi(t), po**
- *carga_w()* => **N, wi**
- *calc_ipc()* => **IPC(t)**

La función ***carga_region()***, cargará información de los datasets dada una región, retornando una tabla con los productos de una región consolidados. La tabla tendrá las series de los productos **pi(t)**.

Estos valores de **pi(t)** deberán ser ponderados por los ponderadores especiales **wi**, los cuales serán cargados por la función ***carga_w()***, que a partir de los ponderadores del INE, retorna la serie especial de la ODEPA y el número de productos **N** de la canasta.

Con **pi(t)** y **wi** como entradas, la función ***calc_ipc()***, retorna una tabla del IPC, donde incluirá el 'IPC-Índice' para la región.

$$IPC_t = \left[\sum_{i=1}^{N} \frac{p_i^t}{p_i^0} w_i \right] 100$$

La función ***carga_región()***, carga individualmente las planillas originales bajadas de la ODEPA a través de la función ***carga_pi()*** y las concatena en una tabla única que es utilizada por la función ***monta_ipc()*** descrita anteriormente.

La función ***carga_pi()*** es una función compuesta que es utilizada para llamar varias otras funciones:

- *limpia_data()*
- *encuentra_variedades()* => **n**
- *precios_promedio()* => **vj**
- *variaciones_precios_promedio()* => **vj(t) / vj(t-1)**

La función ***carga_pi()***:

a) Recibe un dataset específico por categoría de producto de una región dada y limpia los datos a través de ***limpia_data()***;
b) Determina las diferentes variedades dentro de las categorías de producto **n** a través da función ***encuentra_variedades()***;
c) Carga los precios por medio de estos productos, determinando su variación en relación al mes anterior a través de ***variaciones_precios_promedio()***;
d) Calcula la media geométrica entre las variaciones de producto;
e) Retorna entonces una tabla con los valores **pi(t)**.

$$p_i^t = \prod_{j=1}^{n} \left(\frac{v_j^t}{v_j^{t-1}} \right)^{1/t}$$

La función ***limpia_data()***:

a) Carga efectivamente el dataframe a partir de los datos crudos de ODEPA.
b) Busca la celda **Mes/Año**, que es parte de todas las tablas originales y que siempre es la primera celda de la primera línea de los datos. A partir de su posición, hace un "slice" de las líneas superiores (Header) de los datos.
c) Elimina el "Footer", que es la última línea de la tabla.
d) Redefine los nombres de las columnas por los nombres originales de ODEPA
e) Redefine los formatos de las columnas *precios promedio* a "float" y la *fecha* a "datatime".

La función ***encuentra_variedades()***:

a) Encontra las diferentes variedades de productos en un dataset
b) Retorna un conjunto único con las variedades,
c) Retorna el número de variedades distintas **n** para cada producto.

En caso que no existan variedades, como por ejemplo en el caso del producto "Azúcar", la función retornará "n = 1" y "variedades = ['unica']".

La función ***precios_promedio()***:

a) Recibe un dataset trabajado por ***limpia_datos()*** y sus variedades de productos;
b) Retorna un tabla con todos los precios promedio de todas las variedades de producto (estos son los valores promedio originales de ODEPA).
c) En caso no existan datos (**NaN**), la función cargará los datos faltantes con el valor más próximo en el tiempo.
d) Antes de retornar la tabla de precios promedios, llama a la función ***consolida_variedades()***, que:
    a. Verifica que el dataset tenga sub-variedades de productos,
    b. Retorna un dataset consolidado por producto;
    c. Formatea el "index" del dataset por fecha.

La función ***variaciones_precios_promedio()***, trabajará de dos maneras:

a) En caso del parámetro de entrada 'moneda' sea 'False', la función creará una tabla con todas las variaciones de los precios promedio en un dataset relacionado al mes anterior (para todas las **n** variedades). Variaciones pormedio de un mes versus el anterior es utilizado para comparase "un mes contra otro mes".

b) En caso que 'moneda' sea **True**, la función retorna precios nominales y no variaciones. Esto hará con que el valor mensual a ser utilizado en los cálculos sea en unidades monetarias (en el caso Pesos Chilenos) y no la variación del mes corriente sobre el mes anterior. Trabajar derecho con valores nominales es imprtante para se cálcular series de variación del IPC en el tiempo.

6. Librerías Python utilizadas:

- Pandas
- NumPy
- SciPy
- MatPloLlib
- Seaborn
- DateTime
- Scikit-Learn (solo es necesario para ejecutar ML)

7. Referencias

- Site ODEPA: https://www.odepa.gob.cl/precios/precios-al-consumidor-en-linea
- Site INE: http://www.ine.cl/estad%C3%ADsticas/precios/ipc
- Manual Metodológico IPC Base 2013 – INE, Chile