

🖨️🇨🇴 Prueba técnica para optar al puesto de Auditor de riesgo de modelos analíticos. En esta prueba se construyen diferentes modelos predictivos de Machine learning y se elige cual es el que mejores resultados produce.

☆ 0 stars

🔗 0 forks

☆ Star

👁️ Unwatch ▾

<> Code

🕒 Issues

🔗 Pull requests

🎬 Actions

📁 Projects


🛡️ Security

📈 Insights

⚙️ Settings

🔑 master ▾

⋮

 HijoBejuco 1 ...

2 minutes ago ⌚ 15

[View code](#)

# 🖨️ PREDICCIÓN DEL ICV USANDO MACHINE LEARNING 🇨🇴

## 📄 Breve descripción de resultados

- 🇨🇴 En este documento se busca predecir el icv mediante el uso de Machine Learning, para ello, se entrenaron 3 modelos de regresión con los datos *en frecuencia trimestral*, además, el desempeño de cada modelo se calculó usando el **Error Medio Absoluto**, el cual indica **En promedio, las predicciones del modelo difieren 'tantas' veces del valor real**
- 🇨🇴 A continuación se muestra la tabla con los resultados

Tabla 1

	error
RandomForest	0.003747
Linear_Regression	0.011935
DecisionTree	0.006139

- 🇨🇴 De la tabla anterior, se concluye que el modelo con mejor desempeño es **RandomForest()**, ya que tiene el menor valor de error.

## 🇨🇴 Exploración de datos 🇨🇴

### 📄 Resampling y lectural del icv

Inicialmente observamos que la serie de tiempo del índice de cartera vencida (icv) no está en la misma frecuencia temporal que las variables suministradas, puesto que el icv está mensual y las variables están trimestrales.

❗ Por lo anterior, nos vemos obligados a tomar la primer decisión importante en este proyecto: **¿cual de las dos bases de datos modificar?** ya que lo más conveniente es que ambas series de tiempo tengan la misma frecuencia temporal ⌚. Para el caso de las variables, al ser datos macroeconómicos trimestrales, es más difícil convertirlas a mensuales, por lo tanto se decide realizar un **resampling** al icv, para convertirlo de mensual a trimestral.


```
#Read the 'indice de cartera vencida' file, and set the col 0 ('Fecha') be the dataframe index.
indice_df = pd.read_excel('icv_mensual.xlsx', index_col=0)

#Resample the ICV_cartera_total into quarterly periods, so we get the average icv of each three
#months; this is done because the predictor variables are all in quarters frequency, so the
#predictors and the labels must to match.
quarterly_resampled_indice_df = indice_df.ICV_cartera_total.resample('Q').mean()
```

**Figura 1** se carga la información sobre el icv en **indice\_df** y luego se realiza el resampling o cambio de frecuencia temporal a trimestral, y se guarda en el dataframe **quarterly\_resampled\_indice\_df**


**NOTA!**: inicialmente, se entrenarán los modelos con datos trimestrales, ya que así se tendrán más datos y esto mejora la robustez del modelo; luego, más adelante en este documento, se mostrará cómo *se calcula el valor proyectado del icv del sistema financiero a un año*

### Lectura y ajuste de variables

 A continuación se carga la base de datos de variables y se eliminan las 3 primeras filas y la última, esto con el fin de que puedan ser agrupadas correctamente con los datos del icv y formar un sólo dataframe y así explorar más fácil los datos. El objetivo es predecir cual será el icv promedio de los próximos 3 meses usando predictores o variables de la fecha actual.

```
#read the 'variables' dataset
data = pd.read_excel('variables_macro_trimestral.xlsx', index_col=0)
#Remove the first 3 and the last rows, to correctly match with the ICV data
variables_df = data.iloc[3:-1].copy()
```

Figura 2


 Luego, se deben unir los datos del ICV con las variables en un solo dataframe, además de agregar una columna numérica incremental para posteriores análisis, también se usa el comando `info()` para conocer los tipos de datos de las columnas y presencia de valores nulos

```
• #Merge the labels (ICV) and the predictors (variables) into a signal dataframe.
variables_df['icv_cartera_total'] = list(quarterly_resampled_indice_df)
#add an incremental column number to see how the variables change with transcurred time
variables_df['count'] = range(0, len(variables_df.index))
#Let's check the column datatypes
variables_df.info()


✓ 3.4s

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 76 entries, 2001-12-01 to 2020-09-01
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Desempleo             76 non-null    float64
1   IPC                   71 non-null    float64
2   TRM                   76 non-null    float64
3   Exportaciones         47 non-null    float64
4   Importaciones         47 non-null    float64
5   PIB                   63 non-null    float64
6   icv_cartera_total     76 non-null    float64
7   count                 76 non-null    int32
dtypes: float64(7), int32(1)
memory usage: 5.0 KB
```

Figura 3

 De la figura 3 se puede inferir que hay 4 columnas con valores nulos que deben ser modificados posteriormente mediante imputación.

### Gráfico de correlaciones entre variables

 Usando el comando `sns.pairplot()` se genera el siguiente gráfico, el cual muestra en gráficos de disepersión, la correlación existente entre todas las variables, inclusive el icv.

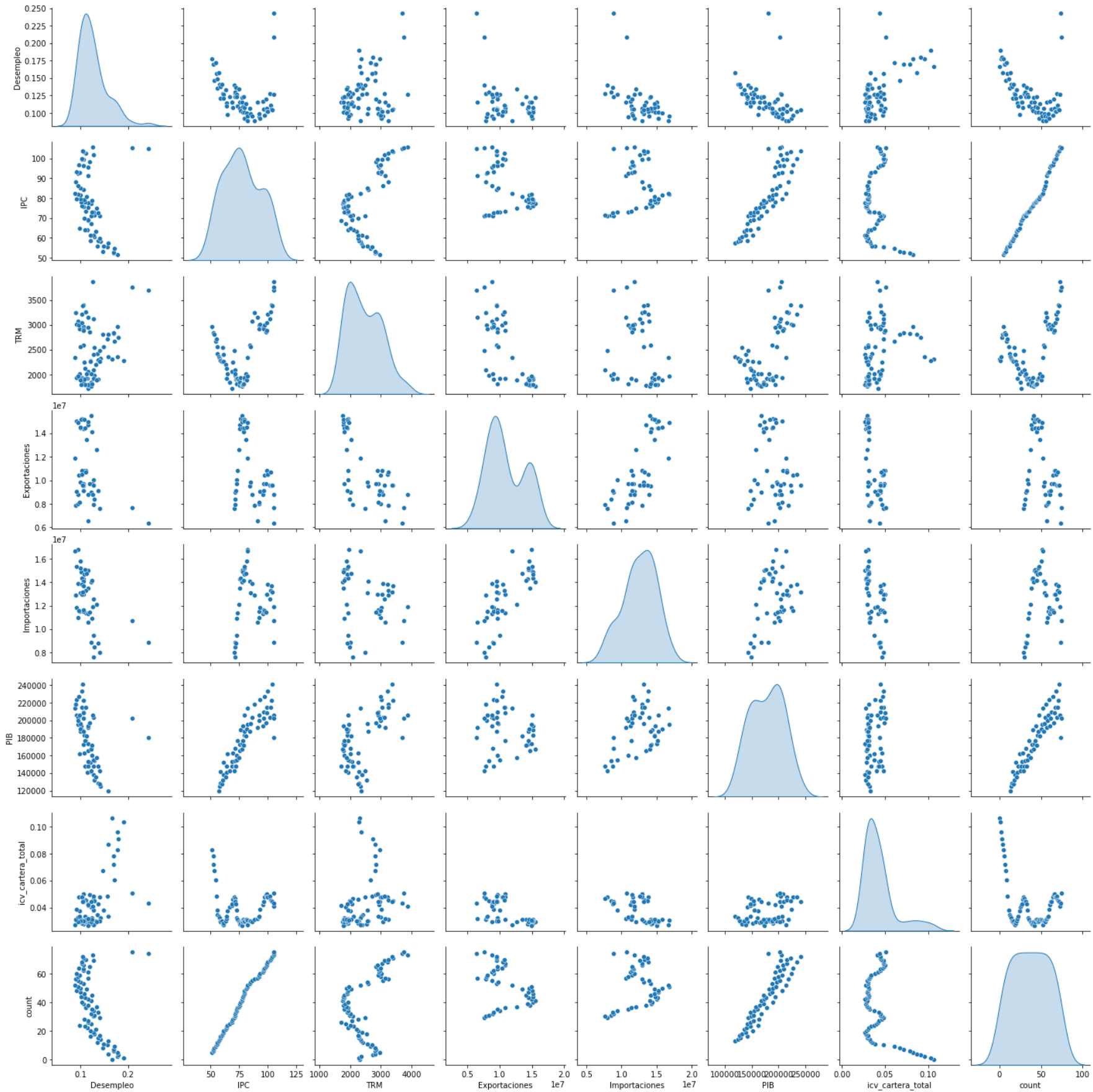


Figura 4

De la figura 4 se observa que la variable `icv_cartera_total` tiene una correlación positiva con la variable **Desempleo**, pero su correlación con las demás variables no es fuerte, hecho que se evidencia más adelante en este proyecto cuando se analicen las relaciones y coeficientes del modelo de regresión lineal, el cual es uno de los varios modelos posteriormente entrenados.

### Imputación usando Regresión lineal

- En la Figura 4 también se puede observar que las variables **PIB** e **IPC** tienen una alta correlación positiva con el tiempo (a medida que pasan los años, el valor del PIB e IPC aumenta); por esta fuerte correlación, se ha tomado la decisión de usar un modelo de Regresión Lineal para reemplazar los datos faltantes en estas dos columnas; para esto, se desarrolló la función `regression_imputer()` la cual hace el proceso de imputación automático.
- El código de la función se muestra a continuación, en dónde básicamente se entrena el modelo con los valores de la columna que no tengan valores nulos y luego de entrenado el modelo, éste se usa para predecir y sustituir los valores nulos por las predicciones realizadas por el modelo.

```
#This function fills the null values of the columns which has high correlation with a column
#which does not have null values
def regression_imputer(dataframe, column_name):
    #dataframe must has a column called 'count' which is only a consecutive number of rows.
    df = dataframe.copy()
    #regression between count and 'column_name'
    #The following code generates a df with the necessary data to train the regression model,
    #excluding the null values.
    model_data = df[['count', f'{column_name}']][df[f'{column_name}'].notnull()]
    #create x and y arrays, we must reshape it because LinearRegression() object only accepts array
    #like inputs
    x = model_data['count'].values.reshape(-1, 1)
    y = model_data[f'{column_name}'].values.reshape(-1, 1)
    #Create the Linear Regression model to estimate the missing data
    regression_model = linear_model.LinearRegression()
    #fit the model
    regression_model.fit(X = x, y = y)
    #generate the x values to be predicted (the 'count' values), the following line extracts from
    #the 'count' column the values where in 'column_name' are nulls, so it extract the X's needed to
    #be predicted.
    x_to_predict = df['count'][df[f'{column_name}'].isnull()].values.reshape(-1,1)
    #Make the predictions
    predictions = regression_model.predict(x_to_predict)
    #We need to transform the predictions array into a numeric list.
    predictions = [float(i) for i in predictions]
    #From the dataframe, select all the null positions in the column, and then replace them by the
    #new predictions
    df[f'{column_name}'][df[f'{column_name}'].isnull()] = predictions
    return df
```

Figura 5

Y a continuación se enseña cómo, usando la función `regression_imputer()` y un ciclo `for`, se reemplazan los valores nulos de las columnas IPC y PIB.

```
#The following list takes the column names which are going to be imputed with regression technique.
columns_to_regression_imputer = ['IPC', 'PIB']

#Create a copy of the dataframe with the null columns
imputed_df = variables_df.copy()

#apply the regression imputer to the columns that need it
for column in columns_to_regression_imputer:

    imputed_df = regression_imputer(imputed_df, column)
```

Figura 6

## Dividir datos entre 'entrenamiento' y 'testeo'

→ Esta etapa es crucial en el desarrollo de cualquier modelo predictivo de Machine Learning, ya que **NUNCA** debe ser usado el 100% de los datos disponibles, sino realizar una partición y utilizar una parte en entrenar el modelo y la otra en testearlo para evaluar éste cómo se comporta haciendo predicciones con datos desconocidos.

**NOTA 1:** esta etapa es fundamental para garantizar la robustez estadística de cualquier modelo

**NOTA 2:** Existen métodos más sofisticados de entrenamiento de modelos (CROSS VALIDATION) y optimización de hiperparámetros (GRIDSEARCHCV), pero debido a la poca cantidad de datos disponibles para entrenar los modelos en este proyecto, se ha optado por una única partición de datos 'entrenamiento' - 'testeo'

→ A continuación, se procede a guardar en `x` la matriz de predictores de nuestros modelos y en `y` el vector de etiquetas o 'labels', en este caso el `icv`, y a partir de estas dos bases de datos, creamos las particiones de 'entrenamiento' y 'testeo' usando la función de `sklearn` `train_test_split()`.

```
#Save the predictors in 'x' and the labels in 'y'
x = imputed_df.drop('icv_cartera_total', axis = 1)
y = list(quarterly_resampled_indice_df)

#create the train and test data
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(x, y, random_state = 42)
```

Figura 7




→ Como último paso antes de entrenar modelos, se deben imputar las columnas 'Exportaciones' e 'Importaciones'; en este caso usaremos los valores del promedio de la columna y la función `SimpleImputer()` de `sklearn`, ver siguiente figura.

```
#Let's impute the columns 'Exportaciones' and 'Importaciones' with their respective means
from sklearn.impute import SimpleImputer
imputer = SimpleImputer()
imputed_train_x = pd.DataFrame(imputer.fit_transform(train_x))
imputed_test_x = pd.DataFrame(imputer.transform(test_x)) #Only we use .transform() because this is
#out_sample data.

#imputation removed col names, put them back
imputed_train_x.columns = train_x.columns
imputed_test_x.columns = test_x.columns
```

Figura 8

## Entrenamiento y evaluación de modelos

→ Se decidió, en esta fase, entrenar y evaluar el desempeño de tres modelos de regresión, los cuales son:  `RandomForestRegressor()`  `LinearRegression()`  `DecisionTreeRegressor()`

→ El desempeño de cada modelo se calculó usando el **Error Medio Absoluto**, el cual indica **En promedio, las predicciones del modelo difieren 'tantas' veces del valor real**. A continuación se muestra un ciclo `for`, el cual itera sobre los 3 modelos y genera el error que cada uno produce evaluado en los datos de 'testeo'

```
models = [RandomForestRegressor(random_state=1)
          , linear_model.LinearRegression()
          , DecisionTreeRegressor(random_state=1)
          ]
errors = []

for model in models:

    predictor = model
    predictor.fit(imputed_train_x, train_y)
    predictions = predictor.predict(imputed_test_x)
    errors.append(mean_absolute_error(test_y, predictions))

pd.DataFrame(errors, ['RandomForest', 'Linear_Regression', 'DecisionTree'], columns = ['error'])
```

✓ 0.9s

	error
RandomForest	0.003747
Linear_Regression	0.011935
DecisionTree	0.006139

Figura 9

## INTERPRETACIÓN COEFICIENTES DEL MODELO DE REGRESIÓN LINEAL

→ Pese a que el modelo de regresión lineal **no fue el modelo que mejor capacidad predictiva tuvo**, éste sin duda, es el modelo que mayor información nos entrega acerca del problema que queremos solucionar, puesto que **el valor de los coeficientes indica el grado de importancia que tiene cada variable a la hora de predecir la variable objetivo**, en este caso el icv. A continuación se muestra el desarrollo en código del modelo de regresión lineal, junto con los valores de los coeficientes de cada variable.



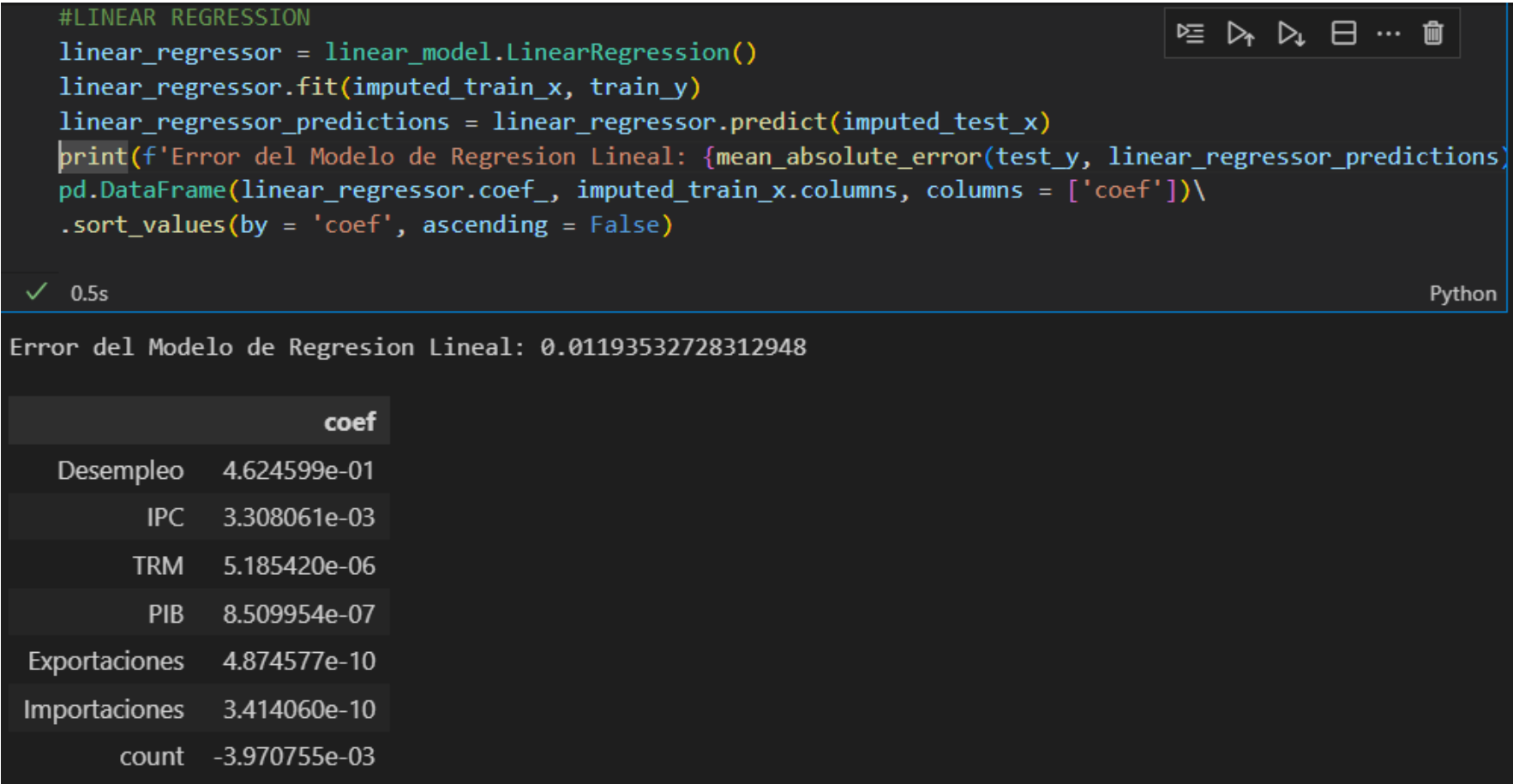


Figura regresión lineal

→ De la figura anterior, se puede inferir que la variable que más influencia tiene a la hora de predecir el icv es **Desempleo**; inclusive, desde un punto de vista simplificado, se podría decir que **es la única variable que tiene un impacto considerable a**

☰ README.md



## Valor proyectado del icv a un año

**NOTA**🔴: El código para la predicción del icv se encuentra en el archivo `icv_anual.ipynb`

- Para realizar la predicción del icv a uno año, se debe realizar un 'resampling' similar al realizado previamente, pero esta vez no trimestral sino anual; adicionalmente este cambio de frecuencia temporal se debe realizar sobre las dos series de tiempo 'icv' y 'variables'.
- El procedimiento de predicciones anuales no contiene nada diferente al procedimiento de predicciones trimestrales, la decisión de haber mostrado el procedimiento de desarrollo de modelos para frecuencias trimestrales es porque **con frecuencias trimestrales disponemos de más datos y por ello los modelos alcanzan mejores desempeños**.
- Para la predicción anual se eligió el modelo `RandomForestRegressor()` porque éste fue el que mejor desempeño tuvo en las predicciones trimestrales; el error encontrado en las predicciones anuales fue **0.022005**, valor mayor al error trimestral, **puesto que se disponen de menos datos para entrenar el modelo**. La construcción del modelo se enseña a continuación.

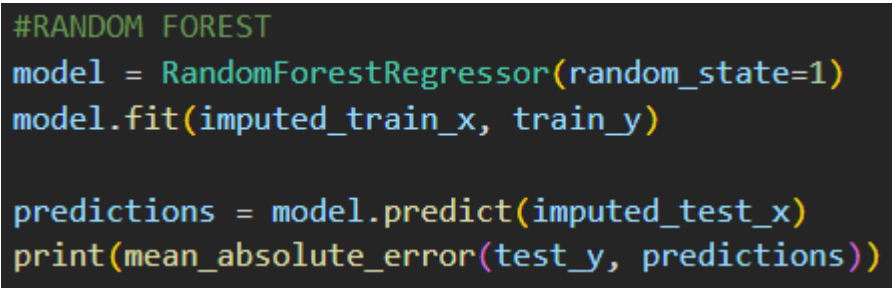


Figura 10

🟢 Para realizar la predicción del icv anual del 2021, se utilizó el siguiente fragmento de código, y el resultado de esta predicción fue de **0.04638 = 4.638%**

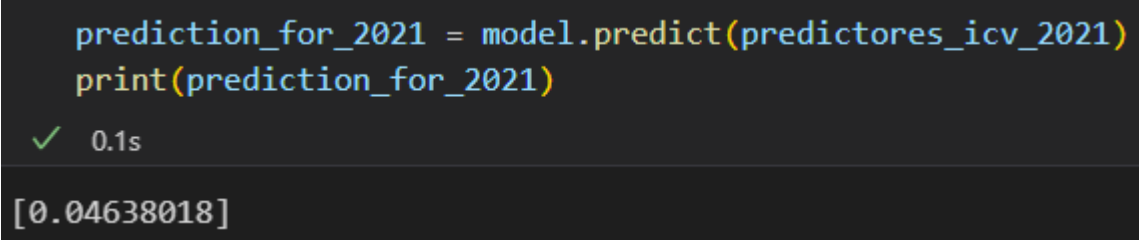


Figura 11

📖 Contrastando el valor predicho del icv con los valores reales, se encuentra poca diferencia, hecho que aumenta la credibilidad de los modelos acá entrenados; el artículo de **la república**, titulado **Estos son los bancos con mayores y menores índices de cartera vencida a julio de 2021** indica que para Julio del 2021 se tenía para Bancolombia un icv de 4,6%, valor similar al predicho por el modelo; pese a que sólo se tiene el valor de los primero 6 meses del icv para el 2021, es un estimativo significativo sobre la efectividad del modelo acá desarrollado.

NOTA🚨: la fuente del artículo de la república, se encuentra en el archivo de texto plano adjunto al proyecto

### 📖Variables adicionales en la predicción del icv👉

📖 A la hora de considerar otras variables con capacidad predictiva para el icv, habrá que tener en cuenta que el icv es una **serie de tiempo** y por ende, se puede aprovechar la **teoría de modelación de series de tiempo (🚨 ARIMA & SARIMA 🚨)**, la cual emplea fuertemente los valores pasados o históricos de la serie de tiempo para predecir los futuros valores. por ejemplo, unas de las herramientas más usadas en series de tiempo son **medias móviles simples** y también **medias móviles exponenciales**.

📖 Por lo anterior, yo recomendaría inferir nuevas variables a partir de los valores pasados del icv, variables como:

- 🌿 media móvil simple
- 🌿 media móvil exponencial
- 🌿 indice anterior
- 🌿 desviación estandard de la serie de tiempo 'n' periodos atrás

🍁 Se pueden inferir **ilimitados posibles predictores en base a los mismos datos históricos**, un ejemplo de uso extensivo de la teoría de predicción de series de tiempo son los modelos de trading financiero, los cuales analizan series de precios y deciden en base a ello tomar alguna decisión en el mercado.

### Releases

No releases published  
[Create a new release](#)

### Packages

No packages published  
[Publish your first package](#)

### Languages

- Jupyter Notebook 100.0%