Universidad Nacional Autónoma de México

Facultad de Ingeniería

PRÁCTICA 13

PRONÓSTICO CON REGRESIÓN LINEAL MÚLTIPLE

ENFOQUE DE APRENDIZAJE SUPERVISADO

Minería de Datos

Profesor:

Dr. Molero Castillo Guillermo Gilberto

Grupo 1

Alumna:

Monroy Velázquez Alejandra Sarahí

No. Cuenta: 314000417

OBJETIVO

Obtener el pronóstico de la saturación de aceite remanente (ROS, Residual Oil Saturation) a partir de las cuatro mediciones de los registros geofísicos convencionales: (RC1) Registro Neutrón, (RC2) Registro Sónico, (RC3) Registro Densidad-Neutrón, y (RC4) Registro Densidad Corregido por Arcilla.

DESARROLLO

El conjunto de datos con el que se trabajará corresponde a mediciones de registros geofísicos convencionales: RC1 (Registro Neutrón), RC2 (Registro Sónico), RC3 (Registro Densidad-Neutrón), y RC4 (Registro Densidad -corregido por arcilla-)

Primero comenzamos la importación de bibliotecas correspondientes que nos ayudarán para la realización del código, las cuales son *pandas* para la manipulación y análisis de datos, *numpy* para crear vectores y matrices, *matplotlib* para la generación de gráficas, así como *seaborn* para la visualización de datos. Por último, la biblioteca files para subir el archivo *csv*.

```
[1]  import pandas as pd                # Para la manipulación y análisis de datos
     import numpy as np                 # Para crear vectores y matrices n dimensionales
     import matplotlib.pyplot as plt    # Para la generación de gráficas a partir de los datos
     import seaborn as sns              # Para la visualización de datos basado en matplotlib
     %matplotlib inline
```

```
[2]  from google.colab import files
     files.upload()
```

Elegir archivos RGeofisicos.csv

- **RGeofisicos.csv**(application/vnd.ms-excel) - 2097 bytes, last modified: 17/11/2021 - 100% done

```
Saving RGeofisicos.csv to RGeofisicos.csv
{'RGeofisicos.csv': b'\xef\xbb\xbfProfundidad,RC1,RC2,RC3,RC4\r\n5660,0.77792425,0.8140290
```

Una vez importadas, el *dataframe* se lee y se despliega en pantalla:

```
[3]  RGeofisicos = pd.read_csv('RGeofisicos.csv')
     RGeofisicos
```

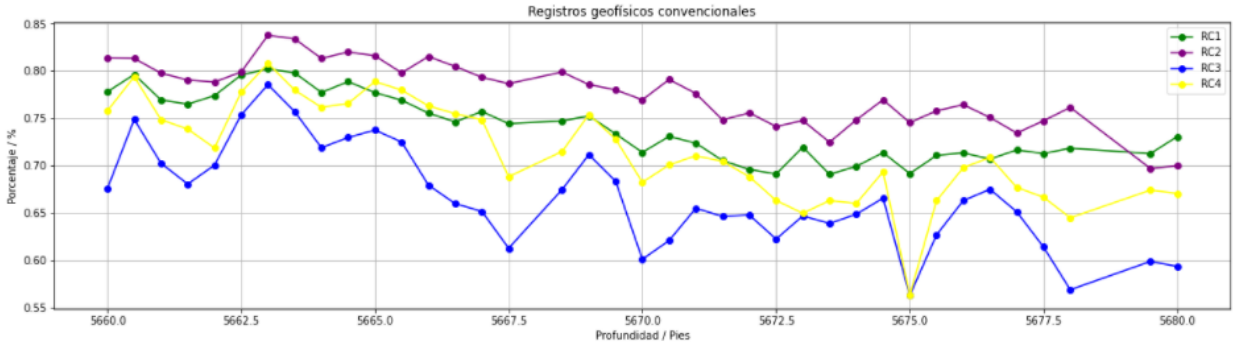
6	5663.0	0.802155	0.837717	0.785441	0.807957
7	5663.5	0.797878	0.833851	0.756847	0.779641
8	5664.0	0.777206	0.813117	0.718713	0.761454
9	5664.5	0.788604	0.820041	0.729582	0.765600
10	5665.0	0.776924	0.815917	0.737350	0.788688
11	5665.5	0.769003	0.797940	0.724736	0.779675
12	5666.0	0.755305	0.815150	0.679189	0.762972
13	5666.5	0.746095	0.804713	0.659602	0.754690
14	5667.0	0.757050	0.793180	0.651374	0.748380
15	5667.5	0.744187	0.786476	0.612430	0.688062
16	5668.5	0.747083	0.798745	0.674513	0.714754
17	5669.0	0.752375	0.785494	0.711418	0.753766
18	5669.5	0.733356	0.779964	0.683226	0.727931
19	5670.0	0.713796	0.769322	0.600747	0.682140
20	5670.5	0.730675	0.790874	0.620547	0.700536

Ahora que el *dataframe* está cargado, comenzamos con los pasos vistos en clase.

Evaluación Visual

Graficamos las mediciones de aceite, incluyendo la profundidad y el porcentajes, para visualizar un panorama general de este dato:

```
[4]  plt.figure(figsize=(20, 5))
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC1'], color='green', marker='o', label='RC1')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC2'], color='purple', marker='o', label='RC2')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC3'], color='blue', marker='o', label='RC3')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC4'], color='yellow', marker='o', label='RC4')
     plt.xlabel('Profundidad / Pies')
     plt.ylabel('Porcentaje / %')
     plt.title('Registros geofísicos convencionales')
     plt.grid(True)
     plt.legend()
     plt.show()
```



Aplicación del algoritmo

Para comenzar a aplicar el algoritmo necesitamos importar la librería *sklearn*, ya que utilizaremos los módulos *linear_model*, *mean_squared_error*, *max_error*, y *r2_score*.

```
[5]  from sklearn import linear_model
     from sklearn.metrics import mean_squared_error, max_error, r2_score
```

Lo siguiente será seleccionar las variables predictoras (X) y la variable a pronosticar (Y):

```
[6]  X_train = np.array(RGeofisicos[['Profundidad', 'RC1', 'RC2','RC3']])
     pd.DataFrame(X_train)
```

6	5663.0	0.802155	0.837717	0.785441
7	5663.5	0.797878	0.833851	0.756847
8	5664.0	0.777206	0.813117	0.718713
9	5664.5	0.788604	0.820041	0.729582
10	5665.0	0.776924	0.815917	0.737350
11	5665.5	0.769003	0.797940	0.724736
12	5666.0	0.755305	0.815150	0.679189
13	5666.5	0.746095	0.804713	0.659602
14	5667.0	0.757050	0.793180	0.651374
15	5667.5	0.744187	0.786476	0.612430
16	5668.5	0.747083	0.798745	0.674513
17	5669.0	0.752375	0.785494	0.711418
18	5669.5	0.733356	0.779964	0.683226

```
[7]  Y_train = np.array(RGeofisicos[['RC4']])
     pd.DataFrame(Y_train)
```

6	0.807957
7	0.779641
8	0.761454
9	0.765600
10	0.788688
11	0.779675
12	0.762972
13	0.754690
14	0.748380
15	0.688062
16	0.714754
17	0.753766
18	0.727931

Las variables predictoras son: *Profundidad*, *RC1*, *RC2*, y *RC3*; mientras que la variable a pronosticar será *RC4*.

Luego, se entrena el modelo a través de Regresión Lineal Múltiple:

```
[8]  RLMultiple = linear_model.LinearRegression()
     RLMultiple.fit(X_train, Y_train)

     LinearRegression()
```

Y se genera el pronóstico:

```
[9]  #Se genera el pronóstico
     Y_pronostico = RLMultiple.predict(X_train)
     pd.DataFrame(Y_pronostico)
```

	0
0	0.747294
1	0.792029
2	0.752073
3	0.737382
4	0.751189
5	0.790661
6	0.818408
7	0.801339
8	0.767461
9	0.780089
10	0.776995

Podemos visualizar una comparativa entre el valor de *RC4* y el pronostico que realizamos, si observamos los valores pronosticados se asemejan a los reales, por lo que esto nos da un indicio de que nuestro modelo se ha entrenado correctamente.

```
[10] RGeofisicos['Pronostico'] = Y_pronostico
     RGeofisicos
```

	Profundidad	RC1	RC2	RC3	RC4	Pronostico
0	5660.0	0.777924	0.814029	0.675698	0.757842	0.747294
1	5660.5	0.796239	0.813167	0.748670	0.793872	0.792029
2	5661.0	0.769231	0.797562	0.702285	0.748362	0.752073
3	5661.5	0.764774	0.790365	0.680289	0.738451	0.737382
4	5662.0	0.773813	0.788184	0.700248	0.718462	0.751189
5	5662.5	0.795627	0.798850	0.753472	0.777537	0.790661
6	5663.0	0.802155	0.837717	0.785441	0.807957	0.818408
7	5663.5	0.797878	0.833851	0.756847	0.779641	0.801339
8	5664.0	0.777206	0.813117	0.718713	0.761454	0.767461
9	5664.5	0.788604	0.820041	0.729582	0.765600	0.780089
10	5665.0	0.776924	0.815917	0.737350	0.788688	0.776995

Obtención de los coeficientes, intercepto, error y score & conformación del modelo de pronóstico

En este paso obtenemos los coeficientes, el intercepto, el error y el score, los cuales nos permitirán realizar el modelo de pronóstico con la siguiente formula:

$$Y = a + b_1X_1 + b_2X_2 \ldots + b_nX_n + u$$

```
[11] print('Coeficientes: \n', RLMultiple.coef_)
     print('Intercepto: \n', RLMultiple.intercept_)
     print("Residuo: %.4f" % max_error(Y_train, Y_pronostico))
     print("MSE: %.4f" % mean_squared_error(Y_train, Y_pronostico))
     print("RMSE: %.4f" % mean_squared_error(Y_train, Y_pronostico, squared=False))
     print('Score (Bondad de ajuste): %.4f' % r2_score(Y_train, Y_pronostico))

     Coeficientes:
      [[-7.50589329e-05  5.06619053e-01  2.27471256e-01  4.89091335e-01]]
     Intercepto:
      [0.26237022]
     Residuo: 0.0684
     MSE: 0.0004
     RMSE: 0.0195
     Score (Bondad de ajuste): 0.8581
```

El modelo queda de la siguiente manera:

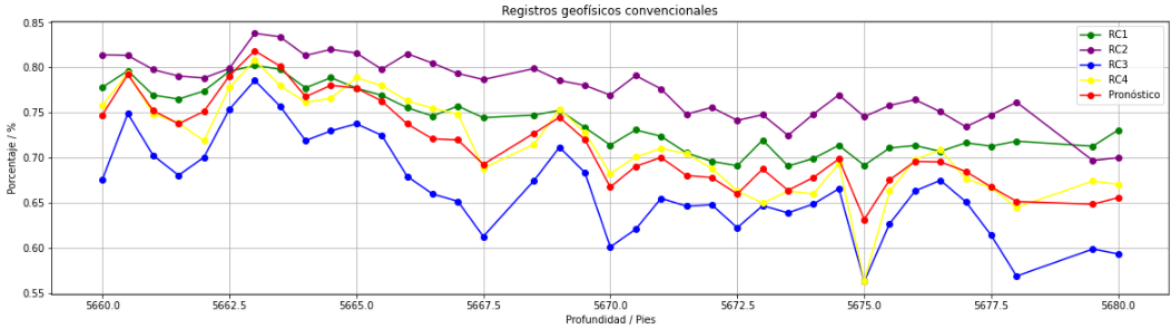
$$Y = 0.2624 - 0.000075(\text{Profundidad}) + 0.5066(RC1) + 0.2275(RC2) + 0.4891(RC3) + 0.0684$$

- Se tiene un Score (Bondad de ajuste) de 0.8581, el cual indica que el pronóstico de la saturación de aceite remanente (SOR), en un determinado nivel de profundidad, se logrará con un 85.81% de efectividad (grado de intensidad).
- Además, los pronósticos del modelo final se alejan en promedio 0.0004 y 0.0195 unidades del valor real, esto es, MSE y RMSE, respectivamente.

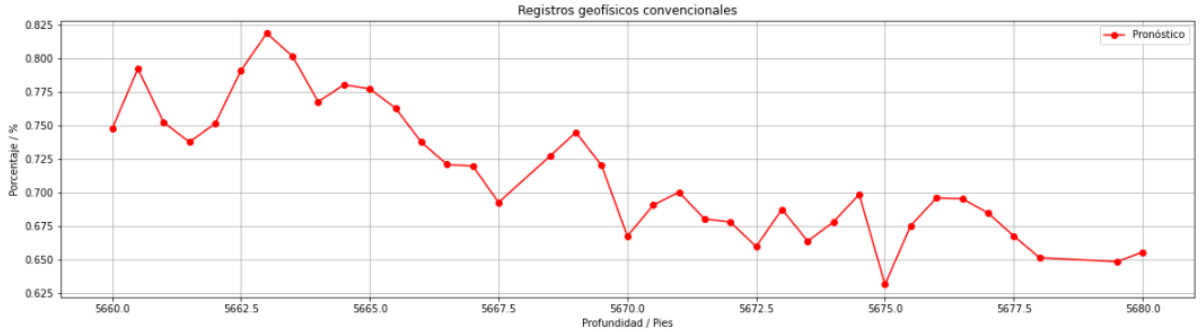
Proyección de los valores reales y pronosticados

Para realizar una comparativa entre los valores reales y los pronosticados, generamos las siguientes gráficas:

```
[12] plt.figure(figsize=(20, 5))
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC1'], color='green', marker='o', label='RC1')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC2'], color='purple', marker='o', label='RC2')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC3'], color='blue', marker='o', label='RC3')
     plt.plot(RGeofisicos['Profundidad'], RGeofisicos['RC4'], color='yellow', marker='o', label='RC4')
     plt.plot(RGeofisicos['Profundidad'], Y_pronostico, color='red', marker='o', label='Pronóstico')
     plt.xlabel('Profundidad / Pies')
     plt.ylabel('Porcentaje / %')
     plt.title('Registros geofísicos convencionales')
     plt.grid(True)
     plt.legend()
     plt.show()
```




```
[13] plt.figure(figsize=(20, 5))
     plt.plot(RGeofisicos['Profundidad'], Y_pronostico, color='red', marker='o', label='Pronóstico')
     plt.xlabel('Profundidad / Pies')
     plt.ylabel('Porcentaje / %')
     plt.title('Registros geofísicos convencionales')
     plt.grid(True)
     plt.legend()
     plt.show()
```



La gráfica marcada en color rojo representa el pronóstico, si miramos con detenimiento es bastante similar a la marcada en color amarillo, por lo que de nuevo esto verifica que nuestro modelo es lo más exacto posible, por lo que podemos confiar en las predicciones que haga.

Nuevos pronósticos

Para nuevos pronósticos implementamos el siguiente bloque de código, donde cada variable predictora tiene un valor; esto servirá para predecir el valor de RC4:

```
[ ]  ROS = pd.DataFrame({'Profundidad': [5680.5], 'RC1': [0.55], 'RC2': [0.64], 'RC3': [0.75]})
     RLMultiple.predict(ROS)

     array([[0.62703853]])
```

CONCLUSIÓN

Así como en la practica anterior, nuevamente implementamos el algoritmo de regresión lineal múltiple, pero ahora aplicado a un set de datos acerca de mediciones convencionales de geofísica. Nuevamente vimos que este algoritmo al ser bien entrenado se convierte en una gran herramienta para predecir datos, lo cual es ideal para la toma de decisiones.