Universidad Nacional Autónoma de México

Inteligencia Artificial

Pronóstico (Integración de valores)

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 8

Profesor: Dr. Guillermo Gilberto Molero Castillo

7 de abril de 2022

Introducción

En esta práctica se generarán pronósticos utilizando regresión lineal múltiple para registros geofísicos sobre la saturación de aceite remanente. Utilizando la regresión lineal múltiple se generará un modelo que pueda predecir RC4 a partir de la profundidad, RC1, RC2 y RC3.

Objetivos

Se tienen mediciones de registros geofísicos convencionales: RC1 (Registro Neutrón), RC2 (Registro Sónico), RC3 (Registro Densidad-Neutrón) y RC4 (Registro Densidad -corregido por arcilla-). Se desea obtener el pronóstico de la saturación de aceite remanente (ROS, Residual Oil Saturation).

Desarrollo

Se importaron todas las bibliotecas relevantes para la realización de la práctica.

```
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
```

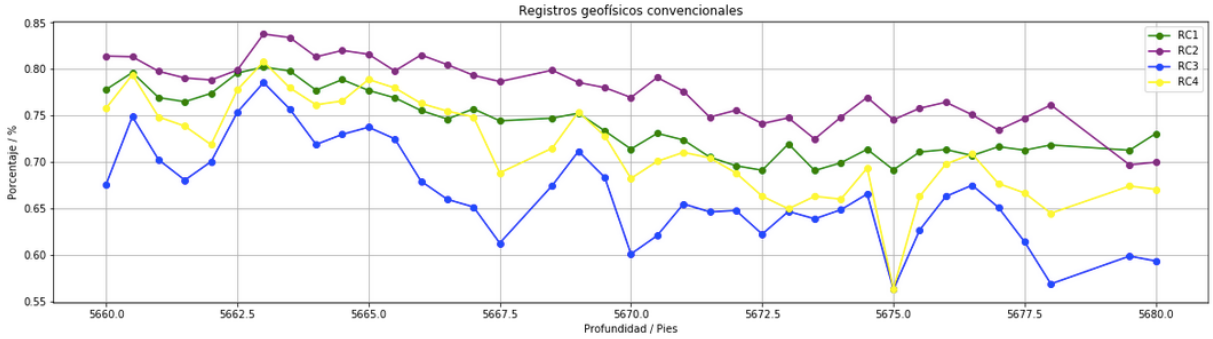
Se muestran los datos donde se puede observar cada profundidad y los diferentes registros que se tienen de un pozo.

```
RGeofisicos = pd.read_csv('RGeofisicos.csv')
RGeofisicos
```

	Profundidad	RC1	RC2	RC3	RC4
0	5660.0	0.777924	0.814029	0.675698	0.757842
1	5660.5	0.796239	0.813167	0.748670	0.793872
2	5661.0	0.769231	0.797562	0.702285	0.748362
3	5661.5	0.764774	0.790365	0.680289	0.738451
4	5662.0	0.773813	0.788184	0.700248	0.718462
5	5662.5	0.795627	0.798850	0.753472	0.777537
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

Utilizando matplotlib se generó una gráfica de la profundidad contra cada uno de los registros. Se puede observar una tendencia general de los diferentes registros, aunque no existe una tendencia muy clara.

```
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()
```



Para generar la regresión lineal múltiple se utilizó el módulo de linear_model de la biblioteca de sklearn. Se importaron otras métricas que se utilizarán para analizar los datos.

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, max_error, r2_score
```

Se generan dos DataFrames con los datos de entrenamiento y se separan entre variables predictoras (x) y la variable a pronosticar. Para el caso de esta práctica se utilizará la profundidad, RC1, RC2 y RC3 como variables predictoras y RC4 como la variable a pronosticar.

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

	0	1	2	3
0	5660.0	0.777924	0.814029	0.675698
1	5660.5	0.796239	0.813167	0.748670
2	5661.0	0.769231	0.797562	0.702285
3	5661.5	0.764774	0.790365	0.680289
4	5662.0	0.773813	0.788184	0.700248

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

	0
0	0.757842
1	0.793872
2	0.748362
3	0.738451

Utilizando la regresión lineal múltiple se entrena el modelo utilizando los datos y se genera el pronóstico a partir del modelo.

```
RLMultiple = linear_model.LinearRegression()
RLMultiple.fit(X_train, Y_train)                    #Se entrena el modelo
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Se genera el pronóstico

```
#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

Se le adjunta a la tabla original la variable de Pronostico.

```
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

Utilizando algunos de los módulos que se importaron al principio se realiza el cálculo de los coeficientes, el intercepto, residuo, MSE, RMSE y la bondad de ajuste.

```
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))  #True devuelve MSE, False devuelve R
print('Score (Bondad de ajuste): %.4f' % r2_score(Y_train, Y_pronostico))
```

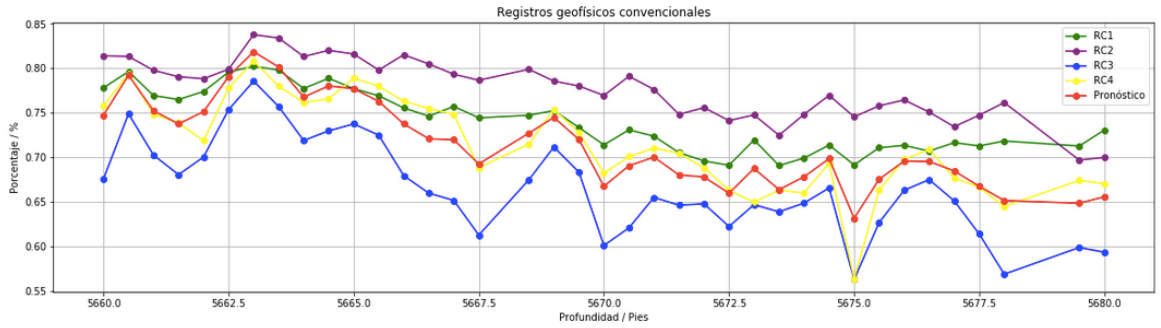
```
Coeficientes:
 [[-7.50589329e-05  5.06619053e-01  2.27471256e-01  4.89091335e-01]]
Intercepto:
 [0.26237022]
Residuo: 0.0684
MSE: 0.0004
RMSE: 0.0195
Score (Bondad de ajuste): 0.8581
```

A partir de los datos anteriores se puede escribir el modelo como:

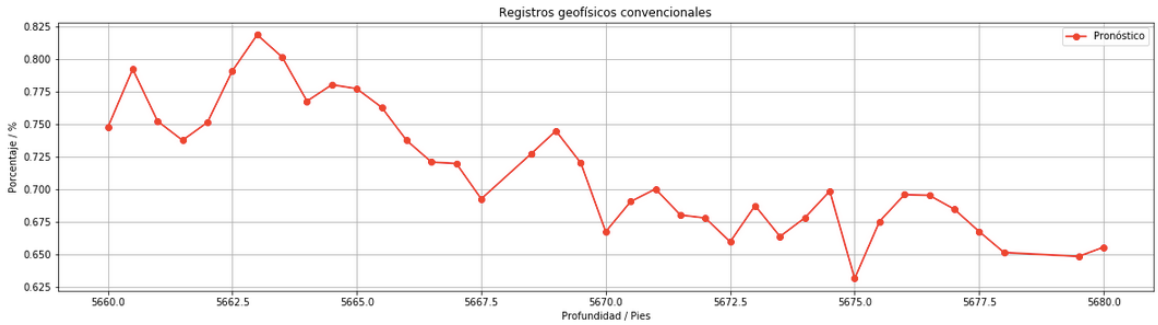
$$Y = 0.2624 - 0.000075(Profundidad) + 0.5066(RC1) + 0.2275(RC2) + 0.4891(RC3) + 0.0684$$

Con la bondad de ajuste se habla de cómo la predicción que se está generando tiene un 85% grado de efectividad. El MSE y RMSE indican cómo los pronósticos finales del modelo se alejan .0004 y .0195 unidades del valor real.

Se genera una nueva gráfica para visualizar los valores pronosticados y qué tan cercanos al valor real se encuentran. El pronóstico es la gráfica roja y RC4 es la amarilla por lo que se puede observar como sí se logró una gráfica cercana a la real excepto en pequeñas partes.



También se generó una gráfica donde solo se encuentran las predicciones para poder visualizar estos datos.



A partir del modelo generado también se pueden generar nuevas predicciones. Para ello se utiliza el método predict y se le debe alimentar al modelo con los cuatro valores que necesita para generar una predicción. Se muestra una predicción de RC4 cuando la profundidad es 5680.5, RC1 es 0.55, RC2 es 0.64 y RC3 es 0.75.

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

```
array([[0.62703853]])
```

Conclusiones

En esta práctica se cumplieron los objetivos y se logró observar una aplicación de la regresión lineal múltiple. Además, se logró implementarla en una Jupyter Notebook utilizando diferentes módulos para obtener un modelo. Para dicho modelo se calculan datos como los coeficientes de la ecuación, su intercepto, su residuo, bondad de ajuste, MSE y RMSE. Cada uno de estos datos se interpretó en el caso específico de los registros de la saturación de aceite remanente en un pozo. Se pudo observar cómo el modelo generado tiene muy buena bondad de ajuste y esto permite tener buenas predicciones a partir del modelo. Con esta práctica se logró generar una regresión múltiple que ayuda a predecir el RC4 a partir de registros geofísicos.