

Prototipo

August 21, 2020

1 Prototipo Delfín

Cargar librerías.

```
[3]: import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import preprocessing
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
    import pandas.util.testing as tm
```

Definir algunas funciones.

1. *formato_datos*: Da formato a los nombres de las columnas del archivo csv.
2. *Eliminar_outliers*: Permite ubicar outliers en un conjunto de datos.
3. *Escalar_datos*: Escalado estándar.

```
[5]: def formato_datos(registro_pozos):
    #Recodificar el nombre de las columnas
    columnas = registro_pozos.columns
    registro_pozos.columns = [str.replace('-', '_') for str in columnas]
    ##cols = ['FI', 'FR', 'IK']
    columnas = ['Profundidad', 'Porosidad', 'Permeabilidad', 'Saturacion_agua',
                'Volumen_arcilla']
    for columna in columnas:
        registro_pozos[columna] = pd.to_numeric(registro_pozos[columna])
    return registro_pozos

def Eliminar_outliers(Conjunto):
    for col in Conjunto.columns:
        col_zscore = str(col) + '_zscore'
        Conjunto[col_zscore] = (Conjunto[col]-Conjunto[col].mean()
                               )/Conjunto[col].std(ddof = 0)
        col_outlier = str(col) + '_outlier'
```

```

    Conjunto[col_outlier] = (abs(Conjunto[col_zscore]) > 3).astype(int)
    #print(Conjunto.col_outlier.value_counts()[1])
    return(Conjunto)

def Escalar_datos (Conjunto):
    escalar = preprocessing.StandardScaler().fit(Conjunto)
    Conjunto_escalado = escalar.transform(Conjunto)
    Conjunto_escalado = pd.DataFrame(Conjunto_escalado)
    return(Conjunto_escalado)

```

Cargar archivo csv de datos.

Queda guardado con el nombre: *registro_pozos*

[6]: `registro_pozos = pd.read_csv('/content/datos_ppp_todos.csv')
print('La dimensión del registro inicial es: ', registro_pozos.shape)`

La dimensión del registro inicial es: (6699, 6)

Existen valores no calculados en el conjunto de datos registrados como 0.00. Estos valores se reemplazan por dato tipo nulo.

[7]: `registro_pozos = registro_pozos.replace({0:np.nan})
print(registro_pozos.info())`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6699 entries, 0 to 6698
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Pozo              6699 non-null   object  
 1   Profundidad       6699 non-null   float64 
 2   Porosidad         6699 non-null   float64 
 3   Permeabilidad    4079 non-null   float64 
 4   Saturacion_agua  6699 non-null   float64 
 5   Volumen_arcilla  6585 non-null   float64 
dtypes: float64(5), object(1)
memory usage: 314.1+ KB
None

```

El informe de los datos indica que faltan valores para *permeabilidad* y *volumen de arcilla*

2 División de datos

En esta sección he creado subconjunto del conjunto *registro_pozos*.

1. *registro_completo*: Contiene las filas que sí contienen información para cada columna del archivo.

```
[8]: registro_completo = registro_pozos.dropna(thresh = 6)
registro_completo = registro_completo.drop_duplicates()
print('La dimensión del registro completo es: ',registro_completo.shape,
      '\n\nInforme del registro completo: \n')
print(registro_completo.info())
#Guardar el registro con el nombre: registro_completo.csv
registro_completo.to_csv('registro_completo.csv')
```

La dimensión del registro completo es: (3965, 6)

Informe del registro completo:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3965 entries, 0 to 6698
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   Pozo              3965 non-null   object  
 1   Profundidad       3965 non-null   float64 
 2   Porosidad         3965 non-null   float64 
 3   Permeabilidad    3965 non-null   float64 
 4   Saturacion_agua  3965 non-null   float64 
 5   Volumen_arcilla  3965 non-null   float64 
dtypes: float64(5), object(1)
memory usage: 216.8+ KB
None
```

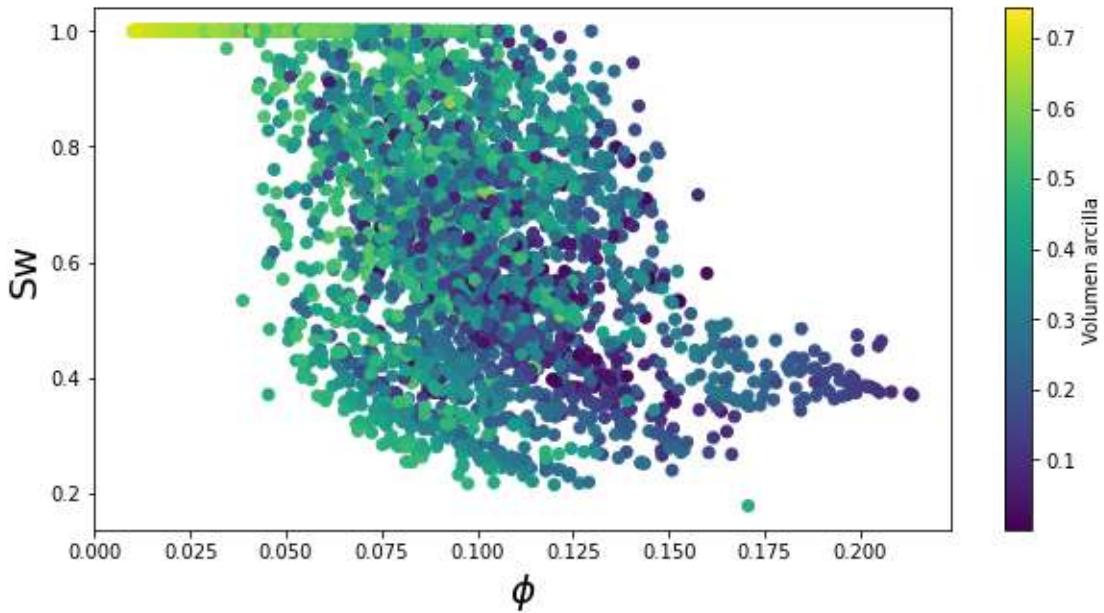
3 Gráficas

Gráfica del registro completo.

Los datos no están escalados.

```
[23]: columnas = ['Porosidad','Saturacion_agua']
registro_pozo = registro_completo[columnas]

plt.figure(figsize = (10,5))
plt.scatter(registro_pozo['Porosidad'], registro_pozo['Saturacion_agua'],
            c = registro_completo['Volumen_arcilla'])
plt.xlabel('$\phi$', fontsize = 20)
plt.ylabel('Sw', fontsize=20)
plt.colorbar(label = 'Volumen arcilla')
plt.show()
```



4 Análisis de componentes principales

He escalado los valores de porosidad usando el método *minmax* de *scikit-learn*.

```
[24]: from sklearn.preprocessing import minmax_scale
PCA = minmax_scale(registro_pozo['Porosidad'])
registro_pozo['Porosidad'] = PCA

plt.figure(figsize = (10,5))
plt.scatter(registro_pozo['Porosidad'], registro_pozo['Saturacion_agua'],
            c = registro_completo['Volumen_arcilla'])
plt.xlabel('$\phi$', fontsize = 20)
plt.ylabel('Sw', fontsize = 20)
plt.colorbar(label = 'Volumen arcilla')
plt.show()
```

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:

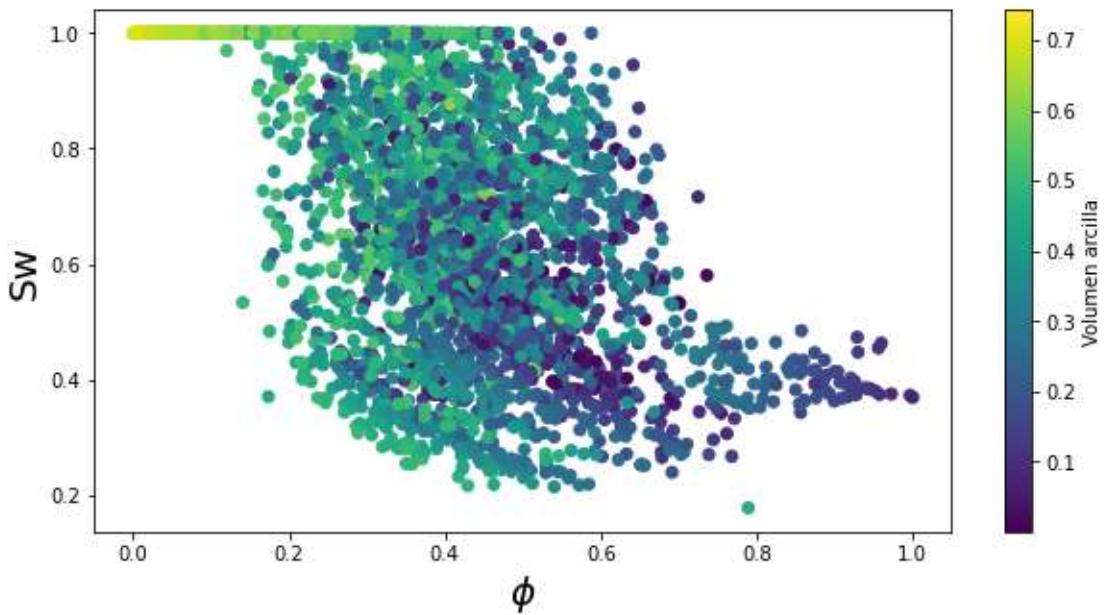
SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

This is separate from the ipykernel package so we can avoid doing imports until



Aplicar PCA para reducir Porosidad e Índice de saturación de agua a 1 dimensión.

[32]: `registro_PCA = registro_pozo`

```
from sklearn.decomposition import PCA

pca = PCA(n_components=1, svd_solver='full')
pca.fit(registro_PCA)

resultados_PCA = pca.transform(registro_PCA)
resultados_PCA = pd.DataFrame(resultados_PCA)

from sklearn.preprocessing import minmax_scale
PCA = minmax_scale(resultados_PCA)
resultados_PCA = pd.DataFrame(PCA)
print('Informe de la dimensión resultante escalada de 0 a 1: \n',
      resultados_PCA.describe())
```

Informe de la dimensión resultante escalada de 0 a 1:

	0
count	3965.000000
mean	0.362898
std	0.254315
min	0.000000
25%	0.114875
50%	0.372998
75%	0.578408
max	1.000000

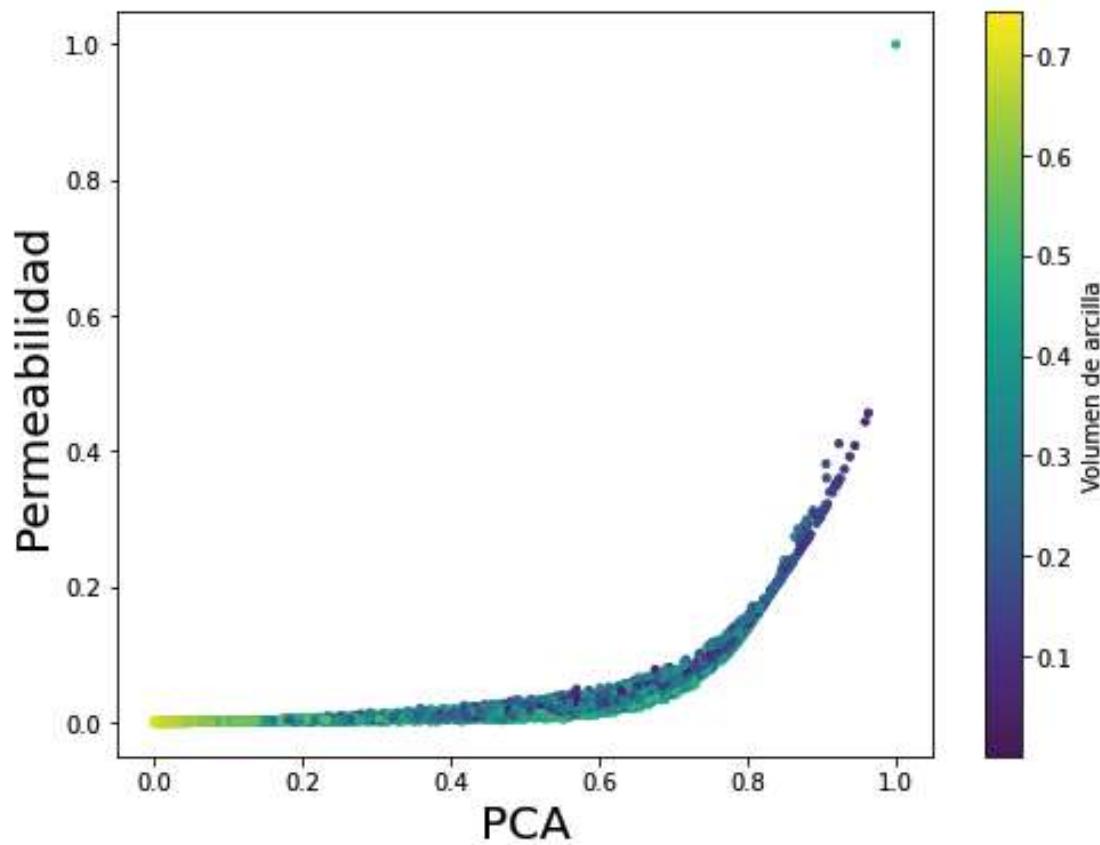
Ahora escalaré la permeabilidad, y usaré la dimensión obtenida en PCA para predecir el volumen de arcilla.

A continuación la gráfica de PCA vs Permeabilidad.

```
[36]: permeabilidad = registro_completo['Permeabilidad']
from sklearn.preprocessing import minmax_scale
K_ = minmax_scale(permeabilidad)
K_ = pd.DataFrame(K_)

resultados_PCA['K'] = pd.DataFrame(K_)

plt.figure(figsize=(8,6))
plt.scatter(resultados_PCA[0], resultados_PCA['K'],
            c = registro_completo['Volumen_arcilla'], s = 10)
plt.xlabel('PCA', fontsize = 20)
plt.ylabel('Permeabilidad', fontsize = 20)
plt.colorbar(label = 'Volumen de arcilla')
plt.show()
print(resultados_PCA.describe())
```



```

count    3965.000000  3965.000000
mean      0.362898    0.025907
std       0.254315    0.054460
min       0.000000    0.000000
25%      0.114875    0.001096
50%      0.372998    0.007369
75%      0.578408    0.024962
max       1.000000    1.000000

```

5 Aplicar SVR

```
[38]: X_MODELO = resultados_PCA
Y_MODELO = registro_completo['Volumen_arcilla']

from sklearn.model_selection import train_test_split
```

Dividir el conjunto en entrenamiento y prueba

```
[39]: X_train, X_test, Y_train, Y_test = train_test_split(X_MODELO,
                                                       Y_MODELO, test_size=0.2)

print('X_Train dimensión: ', X_train.shape)
print('X_Test dimensión: ', X_test.shape)
print('Y_Train dimensión: ', Y_train.shape)
print('Y_Test dimensión: ', Y_test.shape)
```

```

X_Train dimensión: (3172, 2)
X_Test dimensión: (793, 2)
Y_Train dimensión: (3172,)
Y_Test dimensión: (793,)
```

Definir el modelo: SVR

```
[40]: from sklearn import svm

Modelo_SVR = svm.SVR(kernel = 'rbf', C = 90000)
print(Modelo_SVR)
```

```
SVR(C=90000, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
     kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

Entrenar modelo

```
[41]: Modelo_SVR.fit(X_train, Y_train)
```

```
[41]: SVR(C=90000, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
     kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

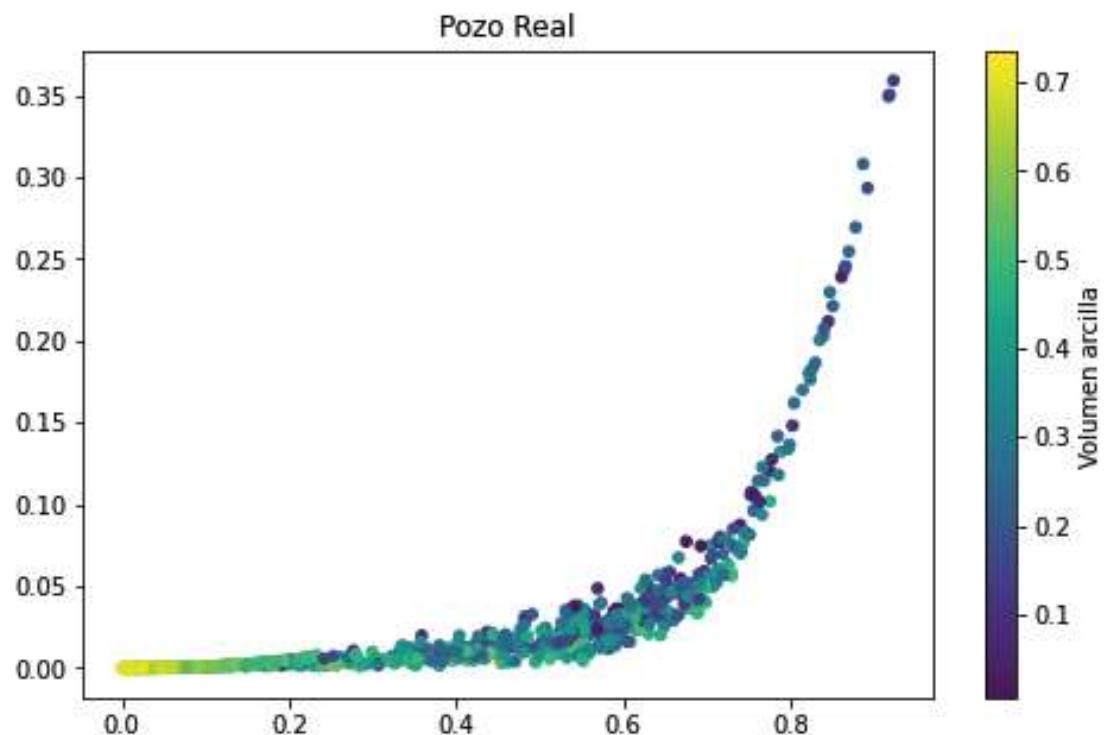
Predecir los datos de prueba

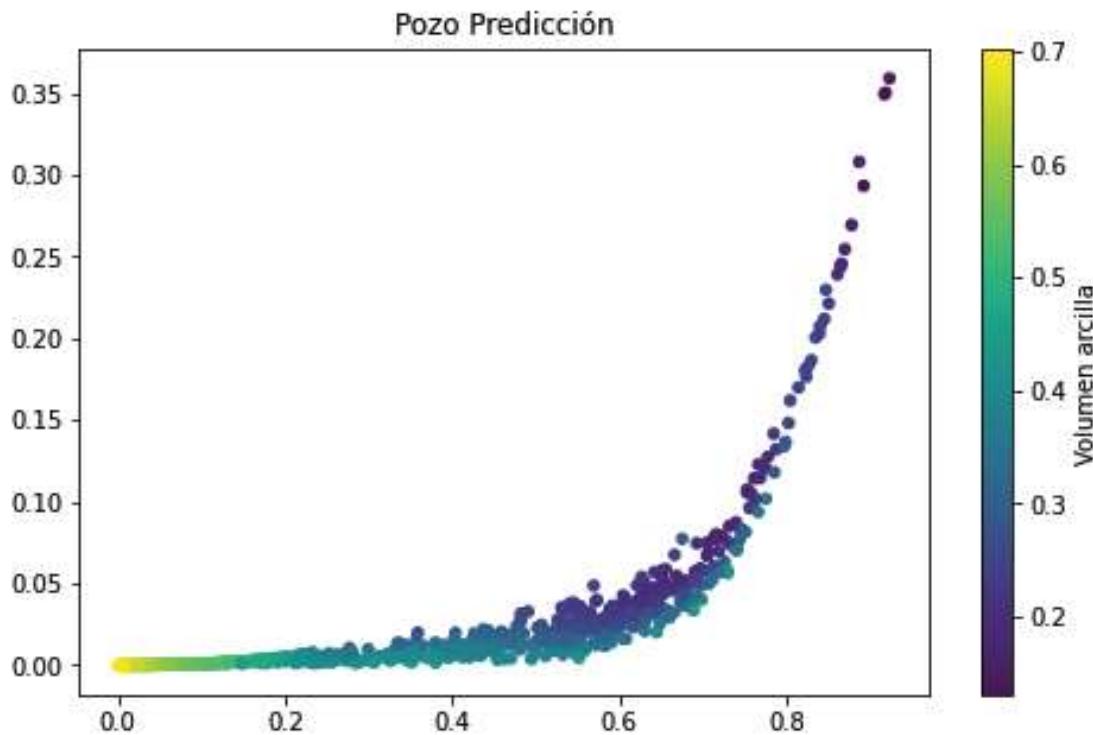
```
[42]: prediccion_SVR_pozo = Modelo_SVR.predict(X_test)
Y_pred = np.round(prediccion_SVR_pozo,4)
```

Comparación de los gráficos: datos reales y de predicción.

```
[45]: plt.figure(figsize=(8,5))
plt.scatter(X_test[0], X_test['K'], c = Y_test, s = 20)
plt.colorbar(label = 'Volumen arcilla')
plt.title('Pozo Real')
plt.show()

plt.figure(figsize=(8,5))
plt.scatter(X_test[0], X_test['K'], c = Y_pred, s = 20)
plt.colorbar(label = 'Volumen arcilla')
plt.title('Pozo Predicción')
plt.show()
```





Evaluar la precisión del modelo

```
[48]: from sklearn.metrics import mean_squared_error
#Usar: MSE
print('MSE: \n')
print(mean_squared_error(Y_test, prediccion_SVR_pozo))
print('Rango de datos reales: ',min(Y_test),max(Y_test))
print('Rango de datos con ML: ',min(Y_pred),max(Y_pred))
print('MAPE: \n')
#Usar: MAPE
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
print(mean_absolute_percentage_error(Y_test,Y_pred))
```

MSE:

0.010049659076969138

Rango de datos reales: 0.0062 0.7349

Rango de datos con ML: 0.1312 0.702

MAPE:

58.24596898384136