OLEOBRÁS

O PROJETO EM QUESTÃO SERÁ CRIADO PARA PREDIZER E AVALIAR OS MELHORES LUGARES PARA A PERFURAÇÃO DE POÇO DE PETRÓLEO. OS DADOS SERÃO IMPORTADOS, INTERPRETADOS E CORRIGIDOS, SE NECESSÁRIO. EM SEGUIDAS, SERÃO APLICADOS TÉCNICAS PARA ALCANÇAR AS RESPOSTAS NECESSÁRIAS.

IMPORTAÇÃO DE BIBLIOTECAS

In [1]:

```
import pandas as pd
import numpy as np


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

BAIXAR O DADOS

In [2]:

```


# Carregar os dados
data_0 = pd.read_csv('/datasets/geo_data_0.csv')
data_1 = pd.read_csv('/datasets/geo_data_1.csv')
data_2 = pd.read_csv('/datasets/geo_data_2.csv')
```

REGIÃO 0

In [3]:

```
data_0.head()
```

Out[3]:

	id	f0	f1	f2	product
0	txEyH	0.705745	-0.497823	1.221170	105.280062
1	2acmU	1.334711	-0.340164	4.365080	73.037750
2	409Wp	1.022732	0.151990	1.419926	85.265647
3	iJLyR	-0.032172	0.139033	2.978566	168.620776
4	Xdl7t	1.988431	0.155413	4.751769	154.036647

In [4]:

```
data_0.describe()
```

Out[4]:

	f0	f1	f2	product
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	0.500419	0.250143	2.502647	92.500000
std	0.871832	0.504433	3.248248	44.288691
min	-1.408605	-0.848218	-12.088328	0.000000
25%	-0.072580	-0.200881	0.287748	56.497507
50%	0.502360	0.250252	2.515969	91.849972
75%	1.073581	0.700646	4.715088	128.564089
max	2.362331	1.343769	16.003790	185.364347

In [5]:

```
data_0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   id       100000 non-null  object 
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
```

In [6]:

```
# valores duplicados
data_0.duplicated().sum()
```

Out[6]:

```
0
```

OBSERVAÇÃO - REGIÃO 0

NÃO POSSUI VALORES AUSENTES, DUPLICADOS OU ERRO DE TIPO DAS COLUNAS.

REGIÃO 1

In [7]:

```
data_1.head()
```

Out[7]:

	id	f0	f1	f2	product
0	kBEdx	-15.001348	-8.276000	-0.005876	3.179103
1	62mP7	14.272088	-3.475083	0.999183	26.953261
2	vyE1P	6.263187	-5.948386	5.001160	134.766305
3	KcrkZ	-13.081196	-11.506057	4.999415	137.945408
4	AHL4O	12.702195	-8.147433	5.004363	134.766305

In [8]:

```
data_1.describe()
```

Out[8]:

	f0	f1	f2	product
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	1.141296	-4.796579	2.494541	68.825000
std	8.965932	5.119872	1.703572	45.944423
min	-31.609576	-26.358598	-0.018144	0.000000
25%	-6.298551	-8.267985	1.000021	26.953261
50%	1.153055	-4.813172	2.011479	57.085625
75%	8.621015	-1.332816	3.999904	107.813044
max	29.421755	18.734063	5.019721	137.945408

In [9]:

```
data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   id       100000 non-null  object 
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
```

In [10]:

```
# valores ausentes
data_1.duplicated().sum()
```

Out[10]:

```
0
```

OBSERVAÇÃO - REGIÃO 1

NÃO POSSUI VALORES AUSENTES, DUPLICADOS OU ERRO DE TIPO DAS COLUNAS.

REGIÃO 2

In [11]:

```
data_2.head()
```

Out[11]:

	id	f0	f1	f2	product
0	fwXo0	-1.146987	0.963328	-0.828965	27.758673
1	WJtFt	0.262778	0.269839	-2.530187	56.069697
2	ovLUW	0.194587	0.289035	-5.586433	62.871910
3	q6cA6	2.236060	-0.553760	0.930038	114.572842
4	WPMUX	-0.515993	1.716266	5.899011	149.600746

In [12]:

```
data_2.describe()
```

Out[12]:

	f0	f1	f2	product
count	100000.000000	100000.000000	100000.000000	100000.000000
mean	0.002023	-0.002081	2.495128	95.000000
std	1.732045	1.730417	3.473445	44.749921
min	-8.760004	-7.084020	-11.970335	0.000000
25%	-1.162288	-1.174820	0.130359	59.450441
50%	0.009424	-0.009482	2.484236	94.925613
75%	1.158535	1.163678	4.858794	130.595027
max	7.238262	7.844801	16.739402	190.029838

In [13]:

```
data_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   id       100000 non-null  object 
 1   f0       100000 non-null  float64
 2   f1       100000 non-null  float64
 3   f2       100000 non-null  float64
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
```

In [14]:

```
# valores duplicados

data_2.duplicated().sum()
```

Out[14]:

```
0
```

OBSERVAÇÃO - REGIÃO 2

NÃO POSSUI VALORES AUSENTES, DUPLICADOS OU ERRO DE TIPO DAS COLUNAS.

TREINO E TESTE DO MODELO PARA CADA REGIÃO

In [15]:

```
def train_and_evaluate_model(data):
    # Separar os recursos (características) e o alvo (volume de reservas)
    features = data.drop(['id', 'product'], axis=1)
    target = data['product']

    # Dividir os dados em conjunto de treinamento e validação (75:25)
    features_train, features_val, target_train, target_val = train_test_split(features, t

    # Treinar o modelo de regressão linear
    model = LinearRegression()
    model.fit(features_train, target_train)

    # Fazer predições para o conjunto de validação
    target_pred = model.predict(features_val)

    # Salvar as predições e respostas corretas para o conjunto de validação
    predictions = pd.DataFrame({'true': target_val, 'predicted': target_pred})

    # Imprimir o volume médio predito de reservas e o REQM do modelo
    avg_predicted_volume = predictions['predicted'].mean()
    rmse = mean_squared_error(target_val, target_pred, squared=False)
    print("Volume médio predito de reservas:", avg_predicted_volume)
    print("REQM (Root Mean Squared Error):", rmse)

    # Retornar as predições e respostas corretas
    return predictions

predictions_0 = train_and_evaluate_model(data_0)
predictions_1 = train_and_evaluate_model(data_1)
predictions_2 = train_and_evaluate_model(data_2)
```

```
Volume médio predito de reservas: 92.3987999065777
REQM (Root Mean Squared Error): 37.756600350261685
Volume médio predito de reservas: 68.71287803913762
REQM (Root Mean Squared Error): 0.890280100102884
Volume médio predito de reservas: 94.77102387765939
REQM (Root Mean Squared Error): 40.14587231134218
```

OBSERVAÇÃO DO TREINAMENTO E TESTE DO MODELO

Volume médio predito de reservas - REGIÃO 0: 92.3987999065777
REQM (Root Mean Squared Error) - REGIÃO 0: 37.756600350261685
Volume médio predito de reservas - REGIÃO 1: 68.71287803913762
REQM (Root Mean Squared Error) - REGIÃO 1: 0.890280100102884
Volume médio predito de reservas - REGIÃO 2: 94.77102387765939
REQM (Root Mean Squared Error) - REGIÃO 2: 40.14587231134218

CÁLCULO DO LUCRO

In [16]:

```
# Armazenar os valores necessários para o cálculo de lucro

budget = 100000000  # Orçamento para o desenvolvimento de 200 poços de petróleo

revenue_per_unit = 4500  # Receita de uma unidade de produto (volume de reservas está em

best_wells= 200 #melhores poços
```

In [17]:

```
# Calcular o volume de reservas suficiente para desenvolver um novo poço sem prejuízos

min_required_volume_0 = budget / (best_wells * revenue_per_unit)
min_required_volume_1 = budget / (best_wells * revenue_per_unit)
min_required_volume_2 = budget / (best_wells * revenue_per_unit)
```

In [18]:

```
# Conclusões sobre a preparação para a etapa de cálculo de lucro
print("Volume de reservas mínimo necessário para desenvolvimento sem prejuízos:")
print("Região 0:", min_required_volume_0)
print("Região 1:", min_required_volume_1)
print("Região 2:", min_required_volume_2)
```

```
Volume de reservas mínimo necessário para desenvolvimento sem prejuízos:
Região 0: 111.11111111111111
Região 1: 111.11111111111111
Região 2: 111.11111111111111
```

FUNÇÃO PARA O CÁLCULO DO LUCRO

In [19]:

```
# Função para calcular o lucro
def calculate_profit(predictions, region_data, print_output=True):
    # Escolher os poços com os valores mais altos de predições
    selected_wells = region_data.loc[predictions.sort_values('predicted', ascending=False

    # Sumarizar o volume alvo de reservas de acordo com essas predições
    target_volume = selected_wells['product'].sum()

    # Calcular o lucro para o volume de reservas recebido
    profit = target_volume * revenue_per_unit - budget

    # Exibir resultados, se necessário
    if print_output:
        if profit > 0:
            print("Região viável para o desenvolvimento de poços de petróleo.")
        else:
            print("Região inviável para o desenvolvimento de poços de petróleo.")
        print("Lucro para o volume de reservas recebido:", profit)
        print("Poços selecionados:")
        print(selected_wells.index[:200])

    # Retornar o lucro e os poços selecionados
    return profit, selected_wells.index[:200]

# Calcular lucro para cada região
profit_0, selected_wells_0 = calculate_profit(predictions_0, data_0)
profit_1, selected_wells_1 = calculate_profit(predictions_1, data_1)
profit_2, selected_wells_2 = calculate_profit(predictions_2, data_2)
```

```
Região viável para o desenvolvimento de poços de petróleo.
Lucro para o volume de reservas recebido: 33591411.14462179
Poços selecionados:
Int64Index([46784, 27658,  6496, 65743, 93716, 29826, 45840, 11404, 84807,
            25827,
            ...
            70346, 14383, 15766, 41388, 85265, 14042, 65925, 65549,  9462,
            39838],
           dtype='int64', length=200)
Região viável para o desenvolvimento de poços de petróleo.
Lucro para o volume de reservas recebido: 24150866.966815114
Poços selecionados:
Int64Index([80439, 14041, 62413, 55563, 42432, 86762, 42661, 50183,   124,
            78924,
            ...
            35487,  7479, 86895, 99899, 86344, 99088,  5058, 80836, 78084,
            77320],
           dtype='int64', length=200)
Região viável para o desenvolvimento de poços de petróleo.
Lucro para o volume de reservas recebido: 25985717.59374112
Poços selecionados:
Int64Index([43931, 84047, 54085, 89165, 64380, 17415,  8317, 54119, 39890,
            71403,
            ...
            20019, 13274, 15214, 66721, 50843, 25434, 68044, 69600, 36778,
            47827],
           dtype='int64', length=200)
```

BOOTSTRAP

In [20]:

```
# Função para calcular lucro usando Bootstrap
def calculate_bootstrap_profit(predictions, region_data):
    profits = []

    for _ in range(1000):
        # Amostrar aleatoriamente as predições
        bootstrap_predictions = predictions.sample(n=500, replace=True)

        # Calcular lucro para a amostra
        profit, _ = calculate_profit(bootstrap_predictions, region_data, print_output=Fal
        profits.append(profit)

    # Calcular lucro médio, intervalo de confiança de 95% e risco de prejuízo
    mean_profit = np.mean(profits)
    ci_lower = np.percentile(profits, 2.5)
    ci_upper = np.percentile(profits, 97.5)
    risk_of_loss = np.mean(np.array(profits) < 0)

    # Apresentar conclusões: sugira uma região para o desenvolvimento de poços de petróle
    if risk_of_loss < 0.025:
        print("Risco de prejuízo inferior a 2.5%.")
        print("Região viável para o desenvolvimento de poços de petróleo.")
    else:
        print("Risco de prejuízo superior a 2.5%.")
        print("Região inviável para o desenvolvimento de poços de petróleo.")
    print("Lucro médio:", mean_profit)
    print("Intervalo de confiança (95%):", (ci_lower, ci_upper))
    print("Risco de prejuízo:", risk_of_loss * 100, "%")

    return mean_profit, ci_lower, ci_upper, risk_of_loss

# Calcular riscos e lucro para cada região usando Bootstrap
mean_profit_0, ci_lower_0, ci_upper_0, risk_of_loss_0 = calculate_bootstrap_profit(predic
mean_profit_1, ci_lower_1, ci_upper_1, risk_of_loss_1 = calculate_bootstrap_profit(predic
mean_profit_2, ci_lower_2, ci_upper_2, risk_of_loss_2 = calculate_bootstrap_profit(predic
```

```
Risco de prejuízo superior a 2.5%.
Região inviável para o desenvolvimento de poços de petróleo.
Lucro médio: 4012768.1663381206
Intervalo de confiança (95%): (-1203423.0218389523, 9366209.30202046)
Risco de prejuízo: 6.9 %
Risco de prejuízo inferior a 2.5%.
Região viável para o desenvolvimento de poços de petróleo.
Lucro médio: 4521407.600600204
Intervalo de confiança (95%): (496876.8683547393, 8545550.82788514)
Risco de prejuízo: 1.2 %
Risco de prejuízo superior a 2.5%.
Região inviável para o desenvolvimento de poços de petróleo.
Lucro médio: 3940598.601101695
Intervalo de confiança (95%): (-1501584.9040720973, 9006615.652055124)
Risco de prejuízo: 7.9 %
```

OS MELHORES 200 POÇOS POR REGIÃO

In [21]:

```
# Selecionar os 200 poços de cada região com base nos riscos
selected_wells_final_0 = data_0.loc[predictions_0.sort_values('predicted', ascending=Fals
selected_wells_final_1 = data_1.loc[predictions_1.sort_values('predicted', ascending=Fals
selected_wells_final_2 = data_2.loc[predictions_2.sort_values('predicted', ascending=Fals

# Exibir os 200 poços selecionados para cada região
print("200 poços selecionados para a Região 0:")
print(selected_wells_final_0)
print("\n200 poços selecionados para a Região 1:")
print(selected_wells_final_1)
print("\n200 poços selecionados para a Região 2:")
print(selected_wells_final_2)
```

```
200 poços selecionados para a Região 0:
          id        f0        f1         f2     product
46784  lfgbR  1.853784 -0.153503  13.585450  153.639837
27658  WcCwe  1.723956 -0.376442  13.139065  140.631646
6496   he3xS  0.370519 -0.283066  13.668868  178.879516
65743  kU92A  0.896968 -0.498996  12.828118  176.807828
93716  2I3WV  1.114191 -0.217015  13.302975  130.985681
...      ...       ...       ...        ...         ...
14042  FbmoM  0.647452 -0.710161   8.595942  132.951877
65925  LagHO  0.724481 -0.555652   8.884751  136.027691
65549  hbgto  0.324662 -0.339168   9.585144  162.142530
9462   h2FcX  0.991234 -0.529205   8.786570  120.536962
39838  wbOXX  1.634711 -0.157964   9.204634  138.424174

[200 rows x 5 columns]

200 poços selecionados para a Região 1:
          id         f0         f1        f2     product
80439  kpPCd -23.884180  -3.773158  5.001008  137.945408
14041  vd9ik -20.401677 -11.205156  5.003276  137.945408
62413  a8qSM -17.391679 -12.347464  5.015414  137.945408
55563  1m8tG -19.352200  -7.969103  5.004618  137.945408
42432  55xmk -19.963551  -3.719157  5.003169  137.945408
...      ...        ...        ...       ...         ...
99088  DcaCI -13.318421  -0.251984  5.002955  137.945408
5058   WgS7g -12.617043  -6.058674  5.002026  137.945408
80836  q7d4S -10.788478 -11.757641  5.007238  137.945408
78084  gKprq -13.109301  -3.913690  5.000987  137.945408
77320  LnEqj -10.867437 -11.071409  5.007265  137.945408

[200 rows x 5 columns]

200 poços selecionados para a Região 2:
          id        f0        f1         f2     product
43931  rtPef -0.214989  1.849141  15.648691  101.225039
84047  KJ6bS  0.517061 -2.686813  15.498363  151.655778
54085  Jz7Ou  1.275966 -2.877779  14.748519   92.947333
89165  yJO5k -0.005191 -1.141924  14.614990   97.775979
64380  bxRO7 -0.800689 -0.417209  14.492273  122.460897
...      ...       ...       ...        ...         ...
25434  S6g0H -2.151721  0.669393  10.761671  179.654566
68044  kLkT4  0.587732 -0.539400  10.790408  142.101687
69600  hFHZS  0.233815 -0.030618  10.772275  121.983862
36778  OCmcT  1.712211 -4.706505  10.762678  184.895101
47827  kMzKw  0.106008  0.177257  10.740001  151.695643

[200 rows x 5 columns]
```

CONCLUSÃO

TRATAMENTO DOS DADOS AS BIBLIOTECAS FORAM IMPORTADAS E OS DADOS FORAM ESTUDADOS. NÃO FOI CONSTATADO VALORES AUSENTES, DUPLICADOS OU ERRO NA TIPAGEM DE ALGUMA COLUNA. PORTANTO, SEGUIMOS AO OBJETIVO.

TREINO E TESTE DO MODELO O RECURSOS FORAM SEPARADOS EM CARACTERÍSTICAS E OBJETIVOS PARA REALIZAR A DIVISÃO DOS CONJUNTOS DE TREINAMENTO E VALIDAÇÃO. o TREINAMENTO FOI REALIZADO COM A REGRESSÃO LINEAR E EM SEGUIDA FIZEMOS AS

PREDIÇÕES:

OBSERVAÇÃO DO TREINAMENTO E TESTE DO MODELO

Volume médio predito de reservas - REGIÃO 0: 92.3987999065777
REQM (Root Mean Squared Error) - REGIÃO 0: 37.756600350261685
Volume médio predito de reservas - REGIÃO 1: 68.71287803913762
REQM (Root Mean Squared Error) - REGIÃO 1: 0.890280100102884
Volume médio predito de reservas - REGIÃO 2: 94.77102387765939
REQM (Root Mean Squared Error) - REGIÃO 2: 40.14587231134218

CÁLCULO DO LUCRO

Orçamento para o desenvolvimento de 200 poços de petróleo = 100 MILHÕES Receita de uma unidade de produto (volume de reservas está em milhares de barris)= 4500

COM OS DADOS FORAM CALCULADOS OS VOLUMES DE RESERVAS SUFICIENTES PARA DESENVOLVER UM NOVO POÇO SEM PREJUÍZO: Região 0: 111.11111111111111
Região 1: 111.11111111111111
Região 2: 111.11111111111111

COM ESTE CÁLCULO EM MÃOS, FOI CRIADO UMA FUNÇÃO QUE RETORNA O LUCRO:

Região 0 viável para o desenvolvimento de poços de petróleo. Lucro para o volume de reservas recebido: 33591411.14462179

Região 1 viável para o desenvolvimento de poços de petróleo. Lucro para o volume de reservas recebido: 24150866.966815114

Região 2 viável para o desenvolvimento de poços de petróleo. Lucro para o volume de reservas recebido: 25985717.59374112

CÁLCULO DO RISCO FOI DESENVOLVIDA UMA FUNÇÃO UTILIZANDO BOOTSTRAP PARA RETORNAR POÇOS COM RISCO MENOR QUE 2.5%, SENDO ASSIM, VIÁVEIS.

AS TRÊS REGIÕES SÃO VIÁVEIS PARA A EXPLORAÇÃO DE PETRÓLEO, POIS ELAS APRESENTAM RISCO DE 0.0%. TODAVIA, APRESENTAM ESTIMATIVAS DIFERENTES:

- A REGIÃO MAIS LUCRATIVA E RECOMENDADA É A REGIÃO 1, POIS É A REGIÃO QUE POSSUI O MENOR RISCO, 1.4%, UM RISCO INFERIOR A 2.5%, E UMA LUCRATIVIDADE ESTIMADA EM CERCA DE 4 MILHÕES.

- AS REGIÕES 0 E 2, POSSUEM UM RISCO DE PREJUÍZO SUPERIOR A 2.5%, RESPECTIVAMENTE POSSUEM UM RISCO DE 6.7% E 7.1% E LUCRATIVIDADE ESTIMADA EM CERCA DE 4 E 3 MILHÕES, TAMBÉM RESPECTIVAMENTE.

OS 500 MELHORES POÇOS DE CADA REGIÃO FORAM ANALISADOS PARA O ESTUDO DE RISCO. DESTES, OS 200 MELHORES FORAM SELECIONADOS PARA O CÁLCULO DE LUCRO, E ESTES 200 MELHORES DE CADA REGIÃO TAMBÉM FORAM EXPOSTOS AO FINAL DO PROJETO, PARA MAIOR COMPREENSÃO E ANÁLISE POR POÇO.