Selection of the optimal location of the drilling for the oil company

Content

Project Description

Project 8 - Selection of the optimal location of the drilling for the oil company

Oil company "GlavGovNeft" would like to drill new hole for oil production.

The provided data has information on three regions with 10 000 potential oil boreholes each, with information on quality and quantity of oil.

Based on the provided information from company it's required to analyze the data and train the model for the prediction of estimated volume of oil in new boreholes and recommend the location of the hole to the oil company where company will get the higher profit. Conduct the analyzis of potential profit ans risks using *Bootstrap*.

To recommend the region it's required to complete the following steps:

- Every region to be studied for potential boreholes, and features of it to be defined;
- The model to be built and quantity of oil products evaluated;
- The bore holes with highest quantity to be selected. Quantity depends on the investment budget and cost of construction of one borehole.
- Reveun of the region is equal to sum of revenue of selected boreholes.

Data import and preparation

In [1]:
```
# libraies import
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import math
```

In [2]:
```
# data loading
try:
    df_1 = pd.read_csv('geo_data_0.csv')
    df_2 = pd.read_csv('geo_data_1.csv')
    df_3 = pd.read_csv('geo_data_2.csv')
except:
    df_1 = pd.read_csv('/datasets/geo_data_0.csv')
    df_1 = pd.read_csv('/datasets/geo_data_1.csv')
    df_1 = pd.read_csv('/datasets/geo_data_2.csv')
```

data overview

In [3]:
```
df_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 [4]:
```
df_1.head()
```

Out[4]:

	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 [5]:

```
# selection of target and features
features_1, target_1 = df_1[['f0','f1','f2']],df_1['product']
```

In [6]:

```
df_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 [7]:

```
df_2.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]:
```
# selection of target and features
features_2, target_2 = df_2[['f0','f1','f2']],df_2['product']
```

In [9]:
```
df_3.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]:
```
df_3.head()
```

Out[10]:

	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 [11]:
```
# selection of target and features
features_3, target_3 = df_3[['f0','f1','f2']],df_3['product']
```

Conclusions

1) Three datasets were imported;

2) all datasets has 10000 rows 5 columns:

- id of borehole;

- quantity of product in it;
- f0,f1, f2 - features of boreholes

Models training and testing

In [12]:
```
# function declaration for prediction of remaining acerage quantity of product in area
def split (features,target):
    model = LinearRegression()
    features_train,features_valid,target_train,target_valid = train_test_split(
        features,target,test_size=0.25,random_state = 12345)
    model.fit(features_train,target_train)
    model.predict(features_valid)
    prediction = model.predict(features_valid)
    rmse = math.sqrt(mean_squared_error(target_valid,prediction))
    return(prediction, target_valid,prediction.mean(),rmse)
```

In [13]:
```
prediction_1, target_valid_1,prediction_mean_1,rmse_1 = split(features_1,target_1)
```

In [14]:
```
print(prediction_mean_1,'\n',rmse_1)
```

```
92.59256778438035
 37.5794217150813
```

Conclusion on 1 region:

1) Average remaining quantity of product is 92.6

2) rmse - 37,5

In [15]:
```
prediction_2, target_valid_2,prediction_mean_2,rmse_2 = split(features_2,target_2)
```

In [16]:
```
print(prediction_mean_2,'\n',rmse_2)
```

```
68.72854689544602
 0.8930992867756165
```

Conclusion on 2 region:

1) Average remaining quantity of product is 68.7

2) rmse - 0.89

In [17]:
```
prediction_3, target_valid_3,prediction_mean_3,rmse_3 = split(features_3,target_3)
```

In [18]:
```
print(prediction_mean_3,'\n',rmse_3)
```

```
94.96504596800489
 40.02970873393434
```

Conclusion on 1 region:

1) Average remaining quantity of product is 94.9

2) rmse - 40

Preparation to revenue calculation

In [19]:
```
PRICE_PER_UNIT = 450000
```

In [20]:
```
TOTAL_BUDGET = 10000000000
```

In [21]:
```
required_qty_of_units = round(TOTAL_BUDGET / PRICE_PER_UNIT,2)
```

In [22]:
```
required_qty_of_units
```

Out[22]:
```
22222.22
```

Revenue calculation conclusions:

- the required minimal quantity of products for achieving of breakeven point is 22222,22 k barrels

Revenue and risks calculation

In [23]:
```
# function for calculaarion of maximum qty of product
def max_qty (prediction):
    prediction = pd.Series(prediction)
    prediction = prediction.sort_values(ascending=False)
    best = prediction[:199]
    return(best.sum())
```

In [24]:
```
region_list = [target_1,target_2,target_3]
```

In [25]:
```
qty_region = []
for i in range(3):
    qty_region.append(round(max_qty(region_list[i]),2))
```

In [26]:
```
qty_region
```

Out[26]:
```
[36782.39, 27451.14, 37721.27]
```

revenue calculation

In [27]:
```
qty_region=pd.Series(qty_region)
income_region = round(qty_region*PRICE_PER_UNIT,2)
```

In [28]:
```
print('{}:'.format(round(income_region,2)))
```

```
0    1.655208e+10
1    1.235301e+10
2    1.697457e+10
dtype: float64:
```

Conclusions

1) Predictied revenue for the 1 region is 16.55 billion rubles.

2) Predictied revenue for the 2 region is 12.23 billion rubles.

2) Predictied revenue for the 3 region is 16.97 billion rubles.

Risks calculation

In [29]:
```
state = np.random.RandomState(12345)
```

In [30]:
```
# function for calculation risk, average value and lower and upper quantilies of revenue
def bootstrap (data):
    values = []
    for i in range(1000):
        data = pd.Series(data)
        subsample = data.sample(n=500, replace=True, random_state=state)
        quantity = pd.Series(subsample)
        max_quantity = max_qty(quantity)
        income = round(max_quantity*PRICE_PER_UNIT,2)
        values.append(income)
    values = pd.Series(values)
    average = round(values.mean(),2)
    upper = round(values.quantile(0.975),2)
    lower = round(values.quantile(0.025),2)
    risk = (values.where(values<TOTAL_BUDGET).count())/values.count()
    return(lower,upper,average,risk)
```

Calculation of required indecies for region 1

In [31]:
```
lower_1,upper_1,average_1,risk_1 = bootstrap(prediction_1)
```

In [32]:
```
print('\n','average:',average_1,'\n','lower of 95% quantile:',
      lower_1,'\n','upper of 95% quantile:',upper_1,'\n', 'risk:','{}:%'.format(round(risk_1*100,2)))
```

```
 average: 10313636100.97
 lower of 95% quantile: 10096323298.45
 upper of 95% quantile: 10553895508.1
 risk: 0.1:%
```

In [33]:
```
profit_1 = (average_1/TOTAL_BUDGET)-1
print('{}:%'.format(round(profit_1*100,2)))
```

```
3.14:%
```

Calculation of required indecies for region 2

In [34]:
```
lower_2,upper_2,average_2,risk_2 = bootstrap(prediction_2)
```

In [35]:
```
print('\n','average:',average_2,'\n','lower of 95% quantile:',
      lower_2,'\n','upper of 95% quantile:',upper_2,'\n', 'risk:','{}:%'.format(round(risk_2*100,2)))
```

```
 average: 10421880341.14
 lower of 95% quantile: 10039403649.91
 upper of 95% quantile: 10823999937.42
 risk: 1.3:%
```

In [36]:
```
profit_2 = (average_2/TOTAL_BUDGET)-1
print('{}:%'.format(round(profit_2*100,2)))
```

```
4.22:%
```

Calculation of required indecies for region 2

In [37]:
```
lower_3,upper_3,average_3,risk_3 = bootstrap(prediction_3)
```

In [38]:
```
print('\n','average:',average_3,'\n','lower of 95% quantile:',lower_3,
      '\n','upper of 95% quantile:',upper_3,'\n', 'risk:','{}:%'.format(round(risk_3*100,2)))
```

```
 average: 10226693504.68
 lower of 95% quantile: 10047320893.18
 upper of 95% quantile: 10406967032.7
 risk: 1.0:%
```

In [39]:
```
profit_3 = (average_3/TOTAL_BUDGET)-1
print('{}:%'.format(round(profit_3*100,2)))
```

```
2.27:%
```

Selection of best region for investments

In [40]:
```
best_region_list = []
profit = [average_1,average_2,average_3]
risks = [risk_1,risk_2,risk_3]
region_number = [1,2,3]
for i in range(len(risks)):
    if risks[i] < 0.025:
        best_region_list.append(region_number[i])
```

```
best_profit = profit[1]
for l in range(len(best_region_list)):
    if profit[l] > best_profit:
        best_region = best_region_list[l]
        best_profit = profit[l]
    else:
        best_region = best_region_list[1]
        best_profit = profit[1]
print('best region:',best_region, 'best profit:',best_profit )
```

```
best region: 2 best profit: 10421880341.14
```

Conclusion

1) Region No 1

- predicted quantity of barrels in region is 36872.39;
- predicted revenue is 16.55 billion rubles;
- average revenue is 10.3 billion rubles;
- lower of 95% quantile: 10 billion rubles;
- upper of 95% quantile: 10.5 billion rubles;
- risk is 0.1:%
- profit is 3.14%

2) Region No 2

- predicted quantity of barrels in region is 27541.14;
- Predictied revenue for the 2 region is 12.23 billion rubles;
- average revenue is 10.4 billion rubles;
- lower of 95% quantile: 10.03 billion rubles;
- upper of 95% quantile: 10.8 billion rubles;
- risk is 1.3:%
- profit is 4.22%

3) Region No 3

- predicted quantity of barrels in region is 37721.21;
- Predictied revenue for the 3 region is 16.97 billion rubles;

- average revenue is 10.2 billion rubles;
- lower of 95% quantile: 10.04 billion rubles;
- upper of 95% quantile: 10.4 billion rubles;
- risk is 1.0:%
- profit is 2.27%

General Conclusion

1) data was successfully imported and prepared;

2) models for every region was trained and quantity of product predicted.

3) breakeven poin was calculated and is 22222,22 k barrels of oil products.

4) revenue for each region is calculated and are:

- Predictied revenue for the 1 region is 16.55 billion rubles.
- Predictied revenue for the 2 region is 12.23 billion rubles.
- Predictied revenue for the 3 region is 16.97 billion rubles.

5) Using bootstrap function the risk and average revenue were calculated:

- Region 1 risk is 0.1:% , average revenue is 10.3 billion rubles;
- Region 2 risk is 1.3:% , average revenue is 10.4 billion rubles;
- Region 3 risk is 1.0:% , average revenue is 10.2 billion rubles;

6) **Recommendation on selection of region**

- Best region for the future oil production is region No 2;
- Acerage predicted revenue on this region is 10,4 billion rubles;
- Level of risk in this region is less than 2,5%, and profit index is higher than in other regions.