[учебная задача]

Выбор локации для скважины

Предоставлены пробы нефти в трёх регионах: в каждом 10 000 месторождений, где измерили качество нефти и объём её запасов. Нужно построить модель машинного обучения, которая поможет определить регион, где добыча принесёт наибольшую прибыль.

1 Загрузка и подготовка данных

Ввод [1]:

```
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from numpy.random import RandomState
from scipy import stats as st
```

Ввод [2]:

```
df0 = pd.read_csv('geo_data_0.csv')
df1 = pd.read_csv('geo_data_1.csv')
df2 = pd.read_csv('geo_data_2.csv')
```

1.1 Описание данных

id — уникальный идентификатор скважины;

f0, f1, f2 — три признака точек (неважно, что они означают, но сами признаки значимы);

product — объём запасов в скважине (тыс. баррелей).

Ввод [3]:

```
# Порядок действий (функции see()):
# Уточним размер таблицы данных
# Уточним количество уникальных id скважин
# Уточним названия столбцов
# Уточним типы данных
# Посмотрим на данные
# Для более полного описания "разлета" значений воспользуемся методом describe()

def see(df):
    print(df.shape)
    print()
    print(pd.DataFrame(list(df0['id'].unique()))[0].count(), '- уникальных ID')
    print()
    print(df.columns)
    print()
    print(df.info())
    display(df.head())
    display(df.describe())
    print('-------------------------------------------------------')

def see_list(A):
    for x in A:
        see(x) # вызов функции rmse
```

Ввод [4]:

```
df = [df0, df1, df2]
see_list(df)
```

```
 4   product  100000 non-null  float64
dtypes: float64(4), object(1)
memory usage: 3.8+ MB
None
```

	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

Ввод [5]:

```
print('Объем запасов в источнике geo_data_0 - {:.0f}'.format(df0['product'].sum()),' (т
print('Объем запасов в источнике geo_data_1 - {:.0f}'.format(df1['product'].sum()),' (т
print('Объем запасов в источнике geo_data_2 - {:.0f}'.format(df2['product'].sum()),' (т
```

```
Объем запасов в источнике geo_data_0 - 9250000  (тыс. баррелей)
Объем запасов в источнике geo_data_1 - 6882500  (тыс. баррелей)
Объем запасов в источнике geo_data_2 - 9500000  (тыс. баррелей)
```

Выводы:

Типы данных не требуется менять.

Переименновывать столбцы не требуется

Пропуски в данных отсутствуют

Прямое кодирование и масштабирование признаков не требуется

Категориальный признак id не несет в себе полезных значений для обучения модели, поэтому исключим id.

product = target

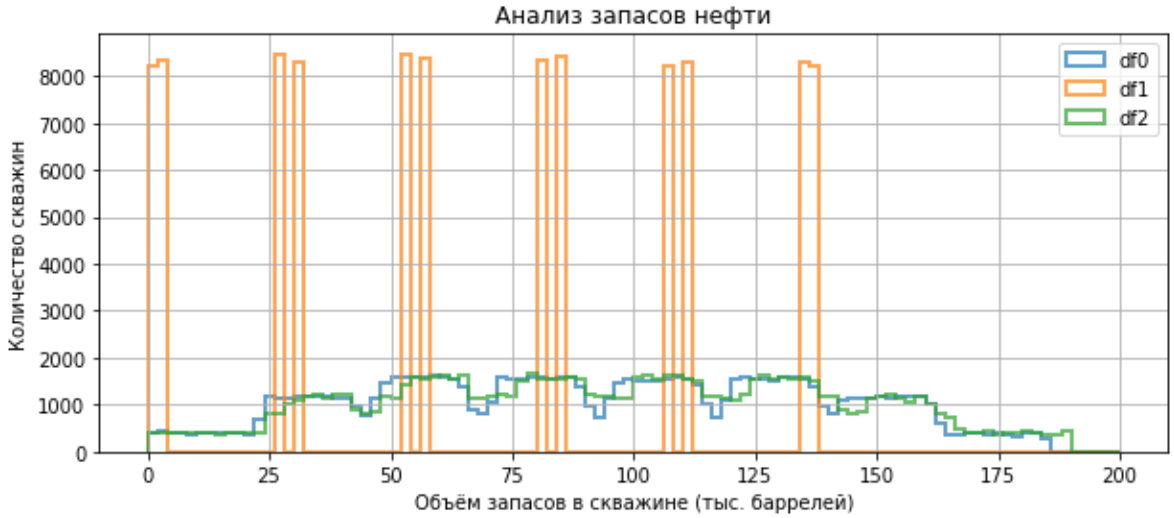
f0, f1, f2 = features

Данных много, поэтому нужен графический анализ данных

Ввод [6]:

```
ax = df0.sort_values(by='product', ascending=False).plot(
    kind='hist',
    y='product',
    histtype='step',
    range=(0, 200),
    bins=100,
    linewidth=2,
    alpha=0.7,
    label='df0',
)
df1.sort_values(by='product', ascending=False).plot(
    kind='hist',
    y='product',
    histtype='step',
    range=(0, 200),
    bins=100,
    linewidth=2,
    alpha=0.7,
    label='df1',
    ax=ax,
)

df2.sort_values(by='product', ascending=False).plot(
    kind='hist',
    y='product',
    histtype='step',
    range=(0, 200),
    bins=100,
    linewidth=2,
    alpha=0.7,
    label='df2',
    ax=ax,
    grid=True,
    legend=True,
    figsize = (10,4)
);
plt.title('Анализ запасов нефти')
plt.xlabel('Объём запасов в скважине (тыс. баррелей)')
plt.ylabel("Количество скважин");
```



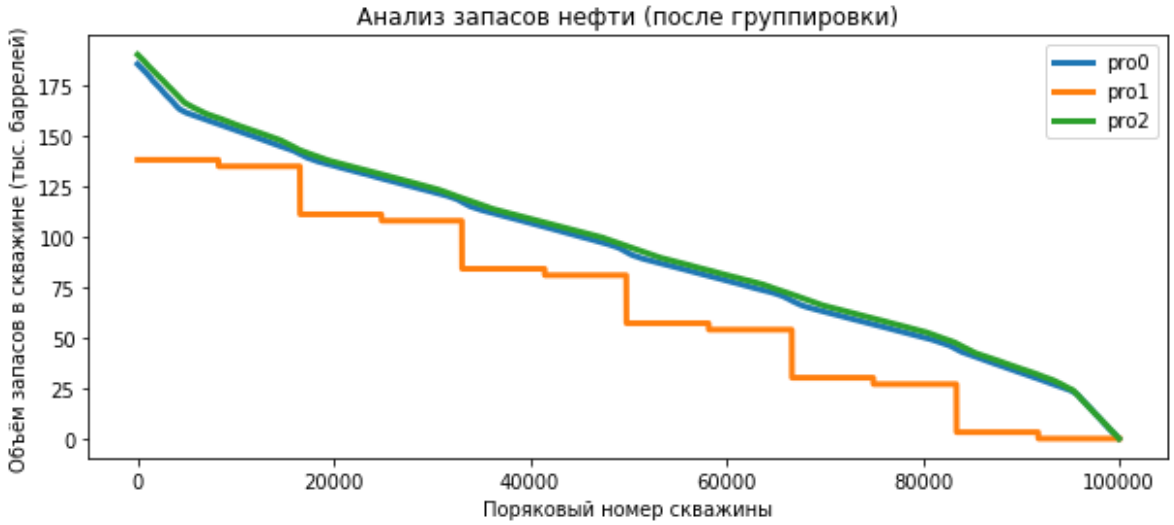
Судя по разпределению значений df1 далёк от нормального распределения.

Попробуем сгруппировать значений "по убыванию" и построить график еще раз.

Ввод [7]:

```
pro_df = pd.DataFrame()
pro_df['pro0'] = df0.sort_values(by='product', ascending=False)['product'].reset_index(
pro_df['pro1'] = df1.sort_values(by='product', ascending=False)['product'].reset_index(
pro_df['pro2'] = df2.sort_values(by='product', ascending=False)['product'].reset_index(

pro_df.reset_index().plot(x='index', y=['pro0', 'pro1', 'pro2'], linewidth=3, figsize =
plt.title('Анализ запасов нефти (после группировки)')
plt.xlabel('Поряковый номер скважины')
plt.ylabel('Объём запасов в скважине (тыс. баррелей)');
plt.show()
```



Вывод:

Про качество нефти здесь нет возможности говорить, но по объему запасов лидирует источник geo_data_2.csv.

Источник geo_data_1.csv содержит данные о сильно похожих друг на друга скважинах, как-будто они искусственно созданы человеком (по ГОСТу), есть такая тенденция добывать нефть на воде и закачивать в искусственные скважины на суше (это в США, а у нас эти скважины называются - резервуары и расположены они на нефтебазах).

Высока вероятность того, что источник geo_data_1.csv - собственные запасы предприятия или очень "искусственно синтезированные данные".

Ввод [8]:

```
# функция создания данных для обучения
def data(df):
    features = df.drop(['id','product'], axis=1)
    target = df['product']
    features_train, features_valid, target_train, target_valid = train_test_split(
    features, target, test_size=0.25, random_state=12345) # 25% данные для валидационно
    return features_train, features_valid, target_train, target_valid
```

Проверим как производится разбивка данных:

Ввод [9]:

```
features_train, features_valid, target_train, target_valid = data(df0)
print('Размер features_train',features_train.shape)
print('Размер target_train',target_train.shape)
print()
print('Размер features_valid',features_valid.shape)
print('Размер target_valid',target_valid.shape)
```

```
Размер features_train (75000, 3)
Размер target_train (75000,)

Размер features_valid (25000, 3)
Размер target_valid (25000,)
```

2 Обучение и проверка модели

Принятая модель - линейная регрессия

Ввод [10]:

```
model = LinearRegression()
```

Ввод [13]:

```
def rmse(df): # таже функция, но с оценкой метрик
    f_train, f_valid, t_train, t_valid = data(df) # вызов функции data
    model.fit(f_train,t_train) # обучение модели на тренировочной выборке
    predictions_valid = model.predict(f_valid) # предсказания модели на валидационной б
    rmse = mean_squared_error(t_valid, predictions_valid)**0.5 # RMSE на валидационной
    print('RMSE модели линейной регрессии на валидационной выборке: {:.3f}'.format(rmse
    print('R2 =', r2_score(t_valid,predictions_valid))
    print('Средний запас предсказанного сырья: {:.2f}(тыс. баррелей)'.format(predictior
    print('----------')

def rmse_list(A):
    for x in A:
        rmse(x) # вызов функции rmse
```

Ввод [14]:

```
rmse_list(df)
```

```
RMSE модели линейной регрессии на валидационной выборке: 37.579
R2 = 0.27994321524487786
Средний запас предсказанного сырья: 92.59(тыс. баррелей)
----------
RMSE модели линейной регрессии на валидационной выборке: 0.893
R2 = 0.9996233978805127
Средний запас предсказанного сырья: 68.73(тыс. баррелей)
----------
RMSE модели линейной регрессии на валидационной выборке: 40.030
R2 = 0.20524758386040443
Средний запас предсказанного сырья: 94.97(тыс. баррелей)
----------
```

Модели обучены, результаты оставляют желать лучшего (ошибка в 40 тыс.баррелей - это очень много).

3 Подготовка к расчёту прибыли

Перейдем к настройке алгоритма решения бизнес-задачи:

1. Синтезируем 1000 раз выборку по 500 скважин

2. Выполним сортировку по убыванию, выберем 200 лучших скважин (сверху) с большим дебетом.

Ввод [15]:

```
state = RandomState(12345) # настроим случайность для Bootstrap
```

4 Расчёт прибыли и рисков

Ввод [16]:

```
BUDGET = 10_000_000_000 # рублей бюджет на разработку скважин в регионе
ONE_POINT = 450_000 # рублей/тыс. баррелей
HOLE_COUNT = 200 # лучших скважин для разработки в регионе
```

Среднее значение дебета скважины для реализации безубыточного сценария (точка безубыточности):

Ввод [17]:

```
BREAKEVEN_POINT = BUDGET / ONE_POINT / HOLE_COUNT # точка безубыточности
print('Желаемый средний объём запасов в скважине {:.1f} (тыс. баррелей)'.format(BREAKE\
ONE_HOLE_BUDGET = BUDGET / HOLE_COUNT # бюджет на разрбаотку одной скважины
print('Бюджет на разработку одной скважины {:.0f} (рублей)'.format(ONE_HOLE_BUDGET))
```

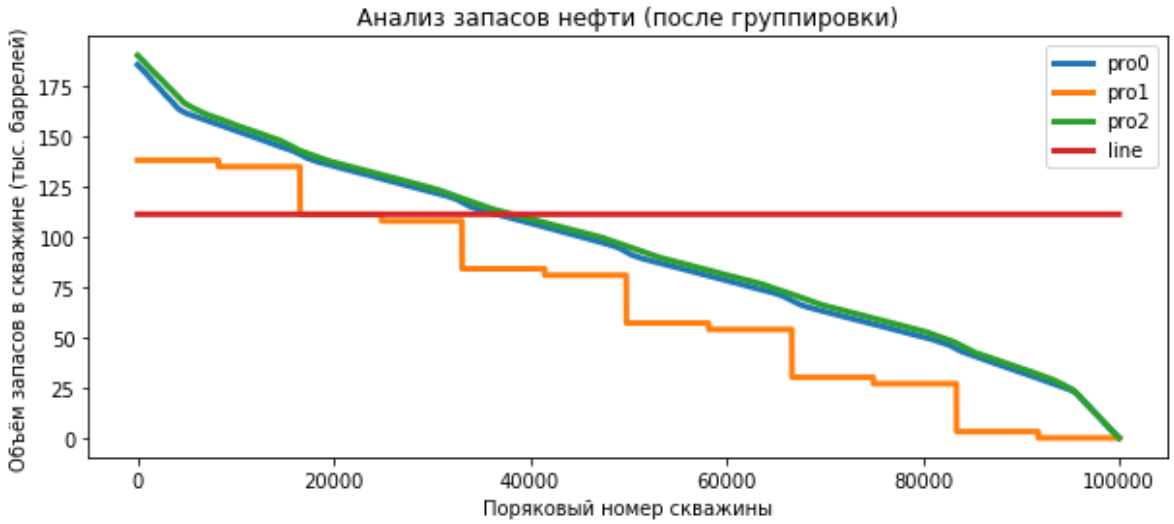
```
Желаемый средний объём запасов в скважине 111.1 (тыс. баррелей)
Бюджет на разработку одной скважины 50000000 (рублей)
```

Добавим к нашим графикам запасов по регионам границу среднего запаса (тыс. баррелей).

Ввод [18]:

```
pro_df['line'] = BREAKEVEN_POINT

pro_df.reset_index().plot(x='index', y=['pro0', 'pro1', 'pro2', 'line'], linewidth=3, f
plt.title('Анализ запасов нефти (после группировки)')
plt.xlabel('Поряковый номер скважины')
plt.ylabel('Объём запасов в скважине (тыс. баррелей)');
plt.show()
```



Рассмотрим 200 независимых исходов событий.

Если в каждом таком независимом исходе будет полученная скважина с значением запаса 111.1 (тыс. баррелей) или выше, то следствием будет прибыль, а не убыток.

Пока отолкнемся от этой посылки попробуем оценить вероятность прибыли (количество хороших исходов), а риск убытков это остальное вероятностное пространство, один минус (вероятность получить прибыльный проект).

Ввод [19]:

```
def test_1000(df): # функция расчета прибыли 1000 раз
    f_train, f_valid, t_train, t_valid = data(df) # вызов функции data
    model.fit(f_train,t_train) # обучение модели на тренировочной выборке
    predictions_valid = model.predict(f_valid)

    values = [] # значение прибыли\убытка

    for _ in range(1000): # создаем тестовую выборку из 200 лучших скважин
        # создаем 500 случайных скважин
        df_predict_500 = pd.Series(predictions_valid).sample(500, random_state=state)
        # вытащим 200 индексов лучших значений
        df_best_predict_200_index = df_predict_500.sort_values(ascending=False).head(20
        # сразу 200 лучших вызову по индексу (здесь не нужно 500 натуральных)
        df_best_valid_200 = t_valid.reset_index(drop=True)[df_best_predict_200_index]
        values.append(df_best_valid_200.sum() * 450_000 - 10_000_000_000)
    return pd.DataFrame(values)
```

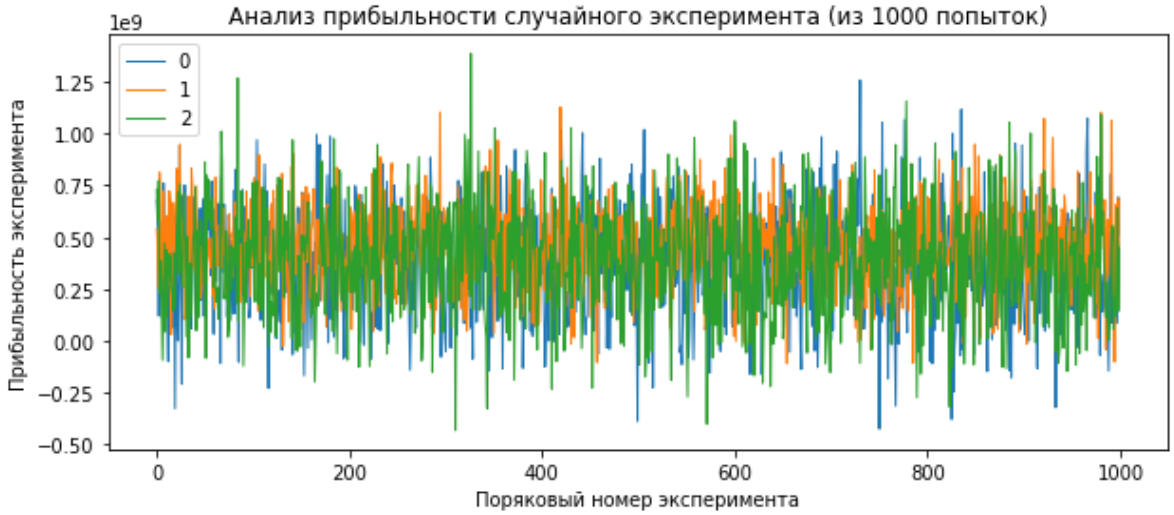
Ввод [20]:

```
revenue_1000_0 = test_1000(df0)
revenue_1000_1 = test_1000(df1)
revenue_1000_2 = test_1000(df2)

revenue = pd.DataFrame()
revenue['0'] = revenue_1000_0
revenue['1'] = revenue_1000_1
revenue['2'] = revenue_1000_2
```

Ввод [21]:

```
revenue.reset_index().plot(x='index', y=['0', '1', '2'], linewidth=1, figsize = (10,4))
plt.title('Анализ прибыльности случайного эксперимента (из 1000 попыток)')
plt.xlabel('Поряковый номер эксперимента')
plt.ylabel('Прибыльность эксперимента');
plt.show()
```



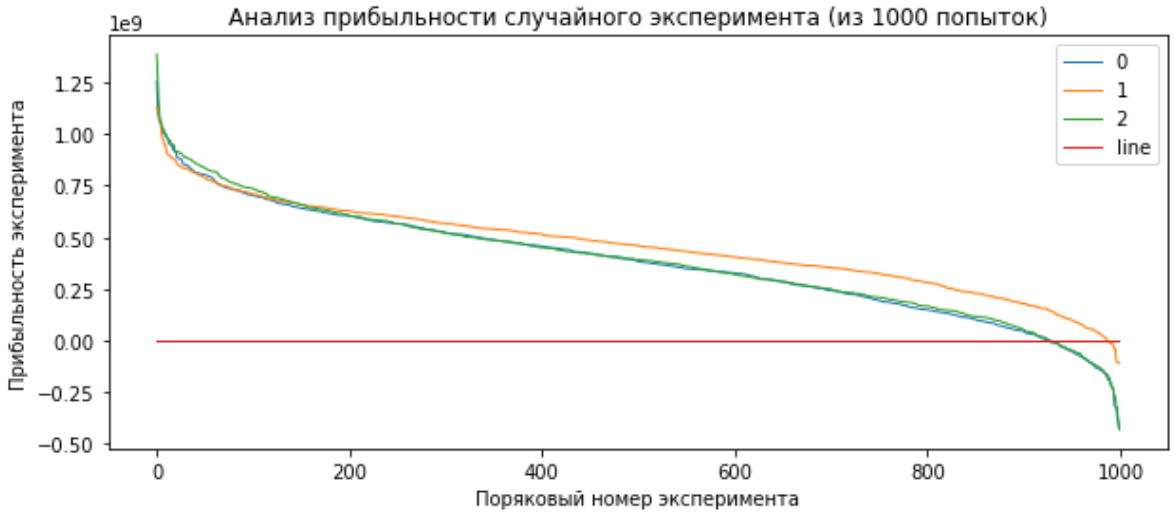
Попробуем выполнить сортировку и показать сколько было прибыльных случаев в регионе 1.

Ввод [22]:

```
revenue_sort = pd.DataFrame()
revenue_sort['0'] = revenue_1000_0.sort_values(by=0, ascending=False).reset_index(drop=
revenue_sort['1'] = revenue_1000_1.sort_values(by=0, ascending=False).reset_index(drop=
revenue_sort['2'] = revenue_1000_2.sort_values(by=0, ascending=False).reset_index(drop=
```

Ввод [23]:

```
revenue_sort['line'] = 0
revenue_sort.reset_index().plot(x='index', y=['0', '1', '2', 'line'], linewidth=1, figs
plt.title('Анализ прибыльности случайного эксперимента (из 1000 попыток)')
plt.xlabel('Поряковый номер эксперимента')
plt.ylabel('Прибыльность эксперимента');
plt.show()
```



Ввод [24]:

```
def interval(df):
    revenue = df[0].mean()

    if  revenue > 0:
        print('Прибыльный регион')
        print('Средняя прибыль {:.0f} руб.'.format(revenue))
    else:
        print('Убыточный регион')
        print('Убыток {:.0f} руб.'.format(revenue))

    print('95%-ый доверительный интервал полученного значения: ',
          df.sort_values(by=0, ascending=False).quantile([0.025, 0.975]).values)

    p1 = df[df[0] >= 0][0].count()/df.shape[0]
    print('Вероятность безубыточного проекта в регионе: {:.2f}'.format(p1))
    print('Вероятность (риск) убыточного проекта в регионе: {:.2f}'.format(1 - p1))
    print()
    print('_________________________')
    print()

def interval_list(A):
    i = 0
    for x in A:
        print('Регион: ', i)
        i += 1
        interval(x) # вызов функции interval
```

Ввод [25]:

```
revenue_1000 = [revenue_1000_0, revenue_1000_1, revenue_1000_2]
interval_list(revenue_1000)
```

```
Прибыльный регион
Средняя прибыль 454785435 руб.
95%-ый доверительный интервал полученного значения:  [[4.67300848e+07]
 [8.40213356e+08]]
Вероятность безубыточного проекта в регионе: 0.99
Вероятность (риск) убыточного проекта в регионе: 0.01

_________________________

Регион:  2
Прибыльный регион
Средняя прибыль 389217074 руб.
95%-ый доверительный интервал полученного значения:  [[-1.15609566e+08]

 [ 9.06512590e+08]]
Вероятность безубыточного проекта в регионе: 0.93
Вероятность (риск) убыточного проекта в регионе: 0.07

_________________________
```

Выводы:

Обучены 3 модели на данных из трех разных регионов.

Каждая модель "наугадывала" себе 1000 раз (исходя из значений признаков f0/f1/f2) по 200 лучших скважин методом bootstrep.

Графическим методом визуализированы прибыльные эксперименты.

Регион 1 имеет самый низкий риск убыточного проекта. К разработке месторождения рекомендован регион 1.