Discount_Impact

April 22, 2021

1 Business Impact of Discoount

We conduct a brief analysis of the strategy that the SME division head proposed. This is not necessarily the optimal strategy. The SME division head proposed that we give a 20% discount to high propensity-to-churn customers. We can assume to start that everyone who is offered a discount will accept it.

1.1 General Workflow

Our task is to calculate the forecast revenue of the set of customers: 1. When no discount is offered, and 2. Whena discount is offered based on some probability cut-off to decide who should receive a discount 20% … and therefore decide where the cut-off should be set so as to maximise revenue.

Do the following:

1.1.1 Load the data

- Load the pickle file of out-of-sample predictions from the best model
- Sort the predictions by predicted probability of churn in descending order (ie, highest predicted probability of churn customers first)

[40]:
```
# imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[41]:
```
# load data
predictions = pd.read_pickle('processed_data/xgb_outOfSamplePredictions.pkl')
```

[42]:
```
predictions.head()
```

[42]:
```
       cons_12m  cons_gas_12m  cons_last_month  forecast_cons_12m  \
5871   5.472865      5.225325         4.229528           3.270241
6997   5.268058      0.000000         4.067220           4.152069
1516   4.218850      0.000000         3.333850           3.131567
11616  4.302374      0.000000         0.000000           3.474324
2532   5.511454      0.000000         4.743518           3.668502

       forecast_discount_energy  forecast_meter_rent_12m  \
```

```
5871                         0.0                 1.811240
6997                         0.0                 2.115976
1516                         0.0                 2.120640
11616                        0.0                 1.287130
2532                         0.0                 1.985382

       forecast_price_energy_p1  forecast_price_energy_p2  \
5871                   0.145711                  0.000000
6997                   0.110955                  0.095842
1516                   0.112860                  0.096521
11616                  0.144149                  0.000000
2532                   0.116509                  0.101397

       forecast_price_pow_p1  has_gas  …  mean_year_price_p2_var  \
5871               44.311378        1  …                0.000000
6997               40.606701        0  …                0.100728
1516               40.606701        0  …                0.099106
11616              44.311378        0  …                0.007124
2532               40.606701        0  …                0.105437

       mean_year_price_p3_var  mean_year_price_p1_fix  mean_year_price_p2_fix  \
5871                 0.000000               44.400265                0.000000
6997                 0.070521               40.579547               24.347725
1516                 0.069847               40.647427               24.388455
11616                0.000000               44.385450                0.000000
2532                 0.075251               40.593123               24.355871

       mean_year_price_p3_fix  mean_year_price_p1  mean_year_price_p2  \
5871                 0.000000           44.550736            0.000000
6997                16.231816           40.700608           24.448453
1516                16.258971           40.767000           24.487562
11616                0.000000           44.533449            0.007124
2532                16.237247           40.718752           24.461308

       mean_year_price_p3  y_test_pred  y_test
5871             0.000000     0.027380       0
6997            16.302337     0.072887       0
1516            16.328818     0.114987       0
11616            0.000000     0.140574       0
2532            16.312498     0.088995       0

[5 rows x 53 columns]
```

1.1.2 Calculate a baseline revenue estimate (no intervention)

Calculate a baseline estimate of the electricity revenue for every customer for the next twelve months based on the forecast consumption and forecast price and actual churn ouotcomoe. Call this `basecase_revenue`

- For customers who end up churning, we should reduce our forecast revenue calculation by 91.9% to account for the customers churn some time between January 2016 and the start of March 2016. (Not knowing when they churn, a reasonable assumption for the lost revenue is the average of 100%, corresponding to churn on 1 January 2016, and 83.9%, corresponding to churn at the end of February, or 59 days into a 365 day year).Call this new variable basecase_revenue_after_churn, ie basecase_revenue_after_churn = basecase_revenue * (1-0.919 * churn)

[43]:
```
# Electricity revenue for each customer consists of energy consumption (amount␣
↪* price) and the meter rent
# The power price may also play a role, but we will ignore it for now
# Note that we need to reverse the log10-trransformation from the data cleaning␣
↪step
predictions['basecase_revenue'] = \
(np.power(10, predictions['forecast_cons_12m'])+1) *␣
↪predictions['forecast_price_energy_p1'] + \
np.power(10, predictions['forecast_meter_rent_12m']+1)
```

[44]:
```
# taking churn into account
predictions['basecase_revenue_after_churn'] = predictions['basecase_revenue'] *␣
↪(1 - 0.919 * predictions['y_test'])
```

1.2 Calculate the estimated benefits and costs of intervention

Now, pick a cut-off probability (eg, 0.5) so that: - Customers with a higher churn probability than cut-off get a discount, and - Customers below the churn-probability get a discount. From this, calculate the revenue of the intervention scenario of this this scenario assuming: - All customers who are offered a discount accept it - Customers who do receive a discount are assumed not to churn in the next twelve months (ie churn probability = 0), and therefore the retained revenue is 0.8 * basecase_revenue, being (1 - discount_frraction) * basecase_revenue - Customers who do not receive a discount are assumed to churn based on the observed dependent variable (ie, a 1 or 0 for whether they actually churned or not)

Now, map out the revenue delta as a function of the cut-off probability in a graph

What cut-off probability approximately optimize the revenue outcome?

Assume for these calculations that the customer does not cinsume more or less electricity because the price changes. (In practice, we would expect that if the customer's cost goes down then their consumption might increase.)

We will see two counterbalancing effects at play: - For true positive we will see revenue retention vs the no-discount scenario - For false positive we will see reduce revenue from giving them a discount when they wouldn't in fact churn

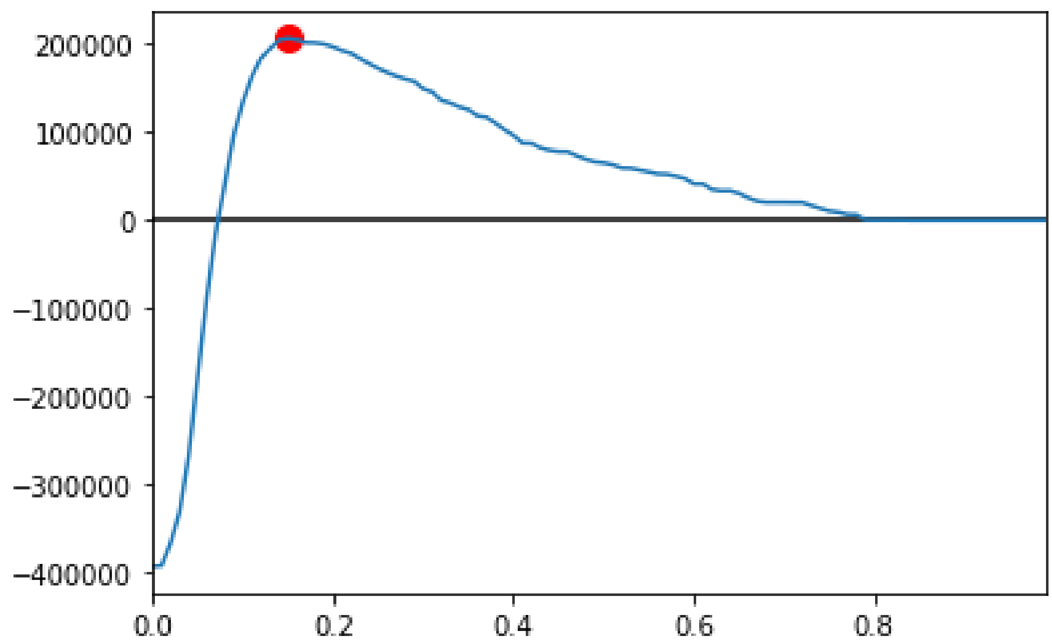
(False negative represent an opportunity cost but not an actual cost differrence between the two scenarios)

The optimal cut-off point will balance the benefits from true positives against the costs of false positives. Our task is to approximately fimd the optimal cut-off point. We many need to make additional assumptions. If we feel the assumptions above aren't justified and that others are better than we should modify our assumptions.

[45]:
```python
def get_rev_delta(pred: pd.DataFrame, cutoff: float = 0.5, discount: float = 0.
 ↪2) -> float:
    '''
    Get the delta of revenue for offering discount for all customers with␣
 ↪predicted churn risk >= cutoff
    '''
    pred['discount_revenue'] = pred['basecase_revenue_after_churn']
    # Churn predicted => discount is given => customer stays for full year,␣
 ↪independent of whether the prediction
    # (false positive, 'free'/unnecessary discount given) or correct
    pred.loc[pred['y_test_pred'] >= cutoff, 'discount_revenue'] =␣
 ↪pred['basecase_revenue'] * (1 - discount)
    # save tthe revenue delta for each custtomer in a separate column
    pred['revenue_delta'] = pred['discount_revenue'] -␣
 ↪pred['basecase_revenue_after_churn']
    return pred['revenue_delta'].sum()
```

[46]:
```python
# Generate a list of possible cutoffs and the corresponding overall revenue␣
 ↪deltas
rev_deltas = pd.Series({cutoff: get_rev_delta(predictions, cutoff = cutoff) for␣
 ↪cutoff in np.arange(0, 1, 0.01)})
```

[47]:
```python
def plot_tradeoff(rev_deltas: pd.Series):
    # Plot the revenue deltas
    rev_deltas.plot()
    # mark optimal point
    max_pred = rev_deltas.idxmax()
    plt.scatter(max_pred, rev_deltas.loc[max_pred], s = 100, c = 'red')
    # Reference line for break-even
    plt.hlines(0, 0, 1)
    plt.show()
    print(f'Maximum benefit at cutoff {max_pred} with revenue delta of␣
 ↪${rev_deltas.loc[max_pred]:,.2f}')

plot_tradeoff(rev_deltas)
```




```
Maximum benefit at cutoff 0.15 with revenue delta of $206,846.04
```

1.3 Optional Extra: How to select the cut-off?

Above, we decide who to offer the discount to based on the probability cut-off.

In this the optimal strategy? - For instance, we might be offering discounts to customers who are not very profitable, thus worsening our overall margins substantially. For example, if offering a discount makes the customer unprofitable on a net margin basis then we might want to let them churn rather than save them. - Even if we only consider revenue, this strategy might not be optimal from a revenue viewpoint. For instance, we can calculate the expected revenue impact of our strategy and priorities customers for discounts that have a high expected revenue impact. (This means that the probability of churn might be high but they also might be valuable customers).

A general principle here is that we can afford to spend moe on retaining high-value customers because the costs of losing them are higher.

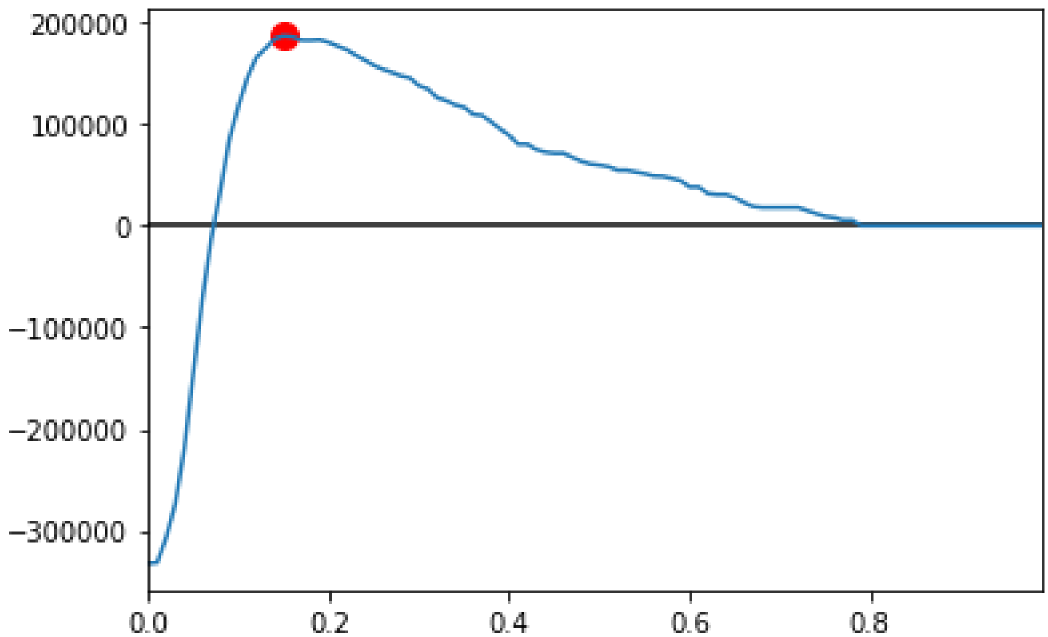
A very common mistake in business applications of churn is to focus on the churn probability whilst forgetting the value impact (to greater or lesswe extents). We have seen many cases where our clients spend as much as effort on retainingupprofitable customers as they do on retaining highly profitable customers.

[48]:
```
def get_rev_delta_high_value(pred: pd.DataFrame, cutoff: float = 0.5, discount:␣
↪float = 0.2, min_rev: float = 500) -> float:
    '''
    Get the delta of revenues for oofering discount for all customers with␣
↪predicted churn risk >= cutoff and revenue
    '''
```

```
    pred['discount_revenue'] = pred['basecase_revenue_after_churn']
    # churn predicted => discount is given for high-value customers => customer␣
 ↪stays for full year, independent
    pred.loc[(pred['y_test_pred'] >= cutoff)&(pred['basecase_revenue'] >␣
 ↪min_rev), 'discount_revenue'] = pred['basecase_revenue'] * (1 - discount)
    # save the revenue delta for each customer in a separate column
    pred['revenue_delta'] = pred['discount_revenue'] -␣
 ↪pred['basecase_revenue_after_churn']
    return pred['revenue_delta'].sum()
```

[49]:
```
# generate a list of possible cutoffs and the corresponding overall revenue␣
 ↪deltas
rev_deltas_high_value = pd.Series({cutoff:␣
 ↪get_rev_delta_high_value(predictions, cutoff = cutoff) for cutoff in np.
 ↪arange(0, 1, 0.01)})
```

[50]:
```
# generate a list of possible cutoffs and the corresponding overall revenue␣
 ↪deltas
plot_tradeoff(rev_deltas_high_value)
```




```
Maximum benefit at cutoff 0.15 with revenue delta of $186,551.34
```

Note: In this case, it doesn't make sense to prioritize large-revenue customers, since the overall revenue delta is much lower than when targeting everyone. However, this is only the case here since the intervention doesn't depend on the number of customers (simply adjusting prices). The interventions usually go beyond simply adjusting prices to prevent churn. There may be the option of intensifying the customer relation, adding key account managers, or other interventions that do incur costs depending on how many customers are targeted. In that case, it may be benefitial to target only a subset of customers to save on these costs, even if the delta in the figure above is reduced.

1.4 Optional Extra: Using forecast rather than actual churn

We may have noticed above that we used actual churn outcomes in calculating the financial impact. Actual churn outcomes are fine if we know them and are conducting a retrospective analysis of the effectiveness of a strategy. This example of analysis is commonly known as 'backtesting', is seeing how well a strategy would have performed historically.

(Of couorse, one must be careful that any analysis is done using out-of-sample data. Conducting the analysis on the training data will lead to predictions that are too optimistic.)

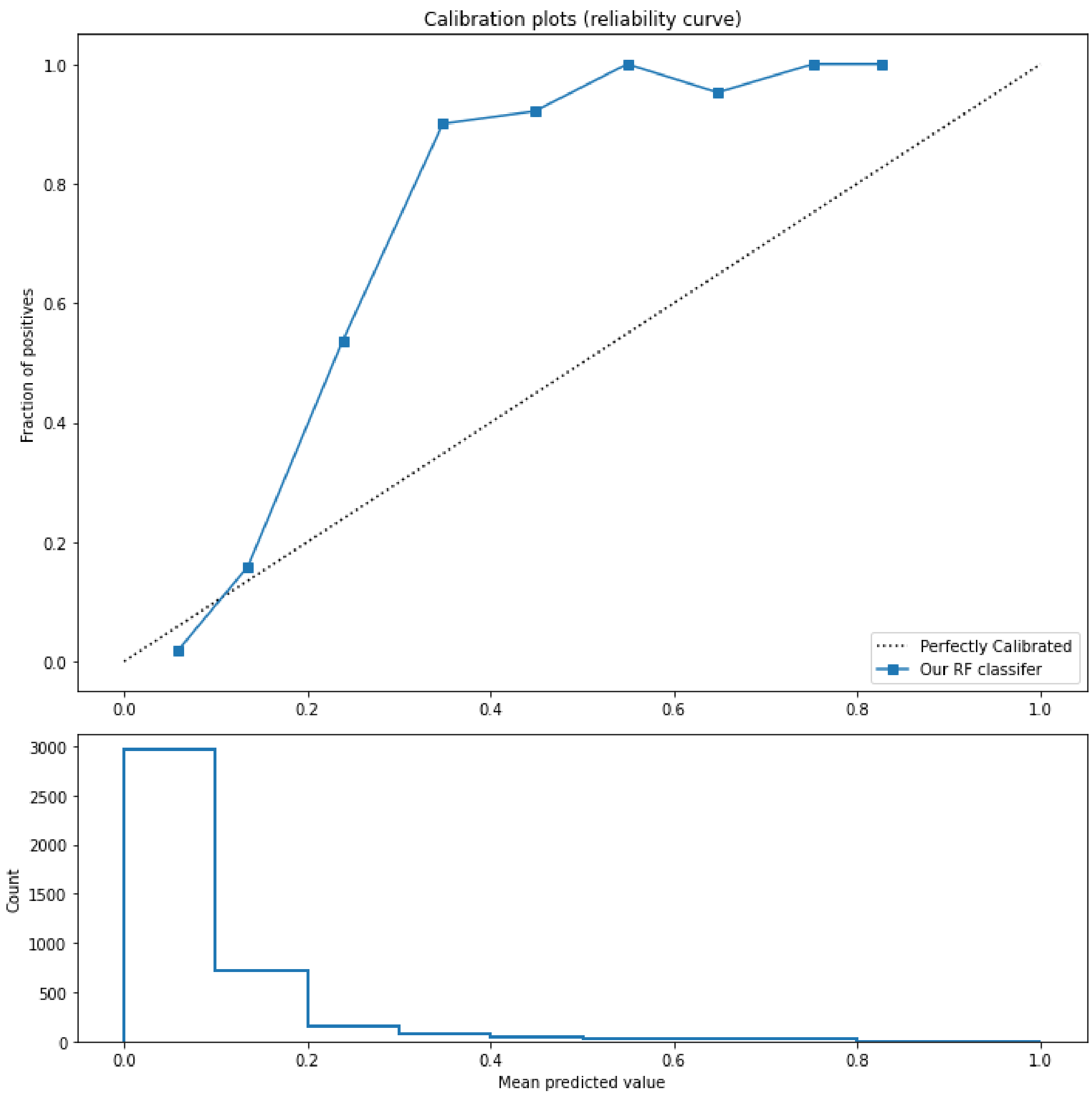
In practive, actual outcomes may not be available because they are in the future.

An alternative is to optimize predicted/forecast revenue based on the probabilities which are on output from our churn model. In this case, we would replace the actual churn outcomes (churn) with the predicted probabiliyt of churn from our model. The results here are obviously model-dependent.

If our model probabilities are poorly calibrated then we can end up with quite poor results from this. Going down thiis path therefore usually requires the extra step of checking how well calibrated the moodel probabilities are, and potentially correcting for any miscalibrating using Platt scaling or isotonic regression.

[51]:
```
# check our calibrating
from sklearn.calibration import calibration_curve

fig = plt.figure(figsize = (10, 10))
ax1 = plt.subplot2grid((3, 1), (0, 0), rowspan = 2)
ax2 = plt.subplot2grid((3, 1), (2, 0))

ax1.plot([0, 1], [0, 1], 'k:', label = 'Perfectly Calibrated')

fraction_of_positives, mean_predicted_value = calibration_curve(y_true =␣
 ↪predictions['y_test'],
                                                               y_prob =␣
 ↪predictions['y_test_pred'],
                                                               n_bins = 10)

ax1.plot(mean_predicted_value, fraction_of_positives, 's-', label = 'Our RF␣
 ↪classifer')
ax2.hist(predictions['y_test_pred'], range = (0, 1), bins = 10, histtype =␣
 ↪'step', lw = 2)

ax1.set_ylabel('Fraction of positives')
ax1.set_ylim([-0.05, 1.05])
ax1.legend(loc='lower right')
ax1.set_title('Calibration plots (reliability curve)')
```

```
ax2.set_xlabel('Mean predicted value')
ax2.set_ylabel('Count')

plt.tight_layout()
```



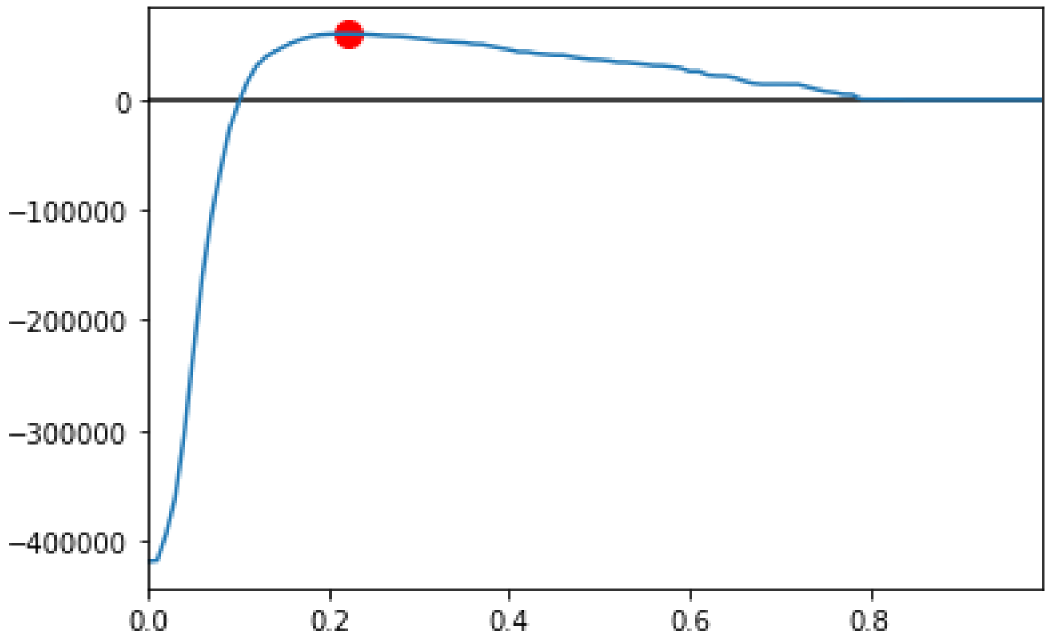
Calibration is OK, but not perfect - let's skip the calibration step here. To use the predicted churn probability, we simply need to replace all 1/0 churn values with it in all calculations.

[52]:
```
# taking churn into account
predictions['basecase_revenue_after_churn'] = predictions['basecase_revenue'] *␣
↪(1 - 0.919 * predictions['y_test_pred'])
```

[53]:
```
def get_rev_delta(pred: pd.DataFrame, cutoff: float = 0.5, discount: float = 0.
↪2) -> float:
    '''
    Get the delta of revenue for offering discount for all customers with␣
↪predicted churn risk >= cutoff
    '''
    pred['discount_revenue'] = pred['basecase_revenue_after_churn']
    # Churn predicted => discount is given => customer stays for full year,␣
↪independent of whether the prediction
    # (false positive, 'free'/unnecessary discount given) or correct
    pred.loc[pred['y_test_pred'] >= cutoff, 'discount_revenue'] =␣
↪pred['basecase_revenue'] * (1 - discount)
    # save tthe revenue delta for each custtomer in a separate column
    pred['revenue_delta'] = pred['discount_revenue'] -␣
↪pred['basecase_revenue_after_churn']
    return pred['revenue_delta'].sum()
```

[54]:
```
rev_deltas = pd.Series({cutoff: get_rev_delta(predictions, cutoff = cutoff) for␣
↪cutoff in np.arange(0, 1, 0.01)})
```

[55]:
```
def plot_tradeoff(rev_deltas: pd.Series):
    # Plot the revenue deltas
    rev_deltas.plot()
    # mark optimal point
    max_pred = rev_deltas.idxmax()
    plt.scatter(max_pred, rev_deltas.loc[max_pred], s = 100, c = 'red')
    # Reference line for break-even
    plt.hlines(0, 0, 1)
    plt.show()
    print(f'Maximum benefit at cutoff {max_pred} with revenue delta of␣
↪${rev_deltas.loc[max_pred]:,.2f}')

plot_tradeoff(rev_deltas)
```




```
Maximum benefit at cutoff 0.22 with revenue delta of $59,680.30
```

[ ]:

[ ]: