# Business Impact of discount

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.

## General workflow

Our task is to calculate the forecast revenue of the set of customers:

1. When no discount is offered, and
2. When a discount is offered based on some probability cut-off to decide who should receive a discount of 20%.

...and therefore decide where the cut-off should be set so as to maximise revenue.

Do the following:

### 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).

```
In [1]:  # Imports
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
```

```
In [2]:  # Load feature matrix + predictions from pickle
         predictions = pd.read_pickle('../processed_data/xgb_outOfSamplePredictions.pkl')
```

### 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 outcome. 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)`

```
In [3]:  # Electricity evenue 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 since we need to ask the client for more d
         # Note that we need to reverse the log10-transformation from the data cleaning step
         predictions['basecase_revenue'] = \
             (np.power(10, predictions['forecast_cons_12m']) + 1) * predictions['forecast_price_energy_p1'] + \
             predictions['forecast_meter_rent_12m']

         # Taking churn into account
         predictions['basecase_revenue_after_churn'] = predictions['basecase_revenue'] * (1 - 0.919 * predictions['churn']
```

### 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 the 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 scenario assuming:

- All customers who are offered a discount accept it
- Customers who do receive a discount are 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_fraction)*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 optimises the revenue outcome?

Assume for these calculations that the customer does not consume 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 postives we will see revenue retention vs the no-discount scenario
- For false positives we will see reduced revenue from giving them a discount when they wouldn't in fact churn.

(False negatives represent an opportunity cost but not an actual cost difference 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 find the optimal cut-off point. We may need to make additional assumptions. If we feel the assumptions above aren't justified and that others are better then we should modify our assumptions.

```
In [4]: def get_rev_delta(pred: pd.DataFrame, cutoff: float=0.5, discount: float=0.2) -> float:
            """
            Get the delta of revenues 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['predict_prob'] >= cutoff, '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()
```

```
In [5]:  # 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)})

         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)
```
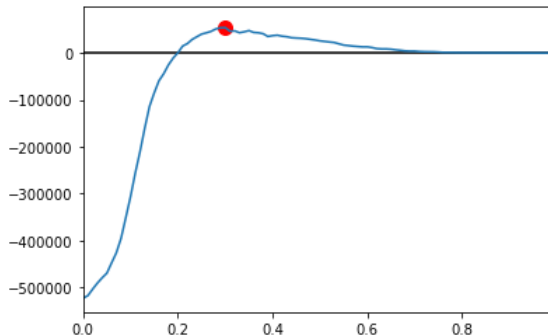


```
Maximum benefit at cutoff 0.3 with revenue delta of $53,985.16
```

## Optional extra: how to select the cut-off?

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

Is 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 prioritise 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 more 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 lesser extents). We have seen many cases where our clients spend as much effort on retaining unprofitable customers as they do on retaining highly profitable customers.

```
In [6]:  def get_rev_delta_high_value(pred: pd.DataFrame, cutoff: float=0.5, discount: float=0.2, min_rev: float=500) -> f
             """
             Get the delta of revenues for offering discount for all customers with predicted churn risk >= cutoff and rev
             """
             pred['discount_revenue'] = pred['basecase_revenue_after_churn']

             # Churn predicted => discount is given for high-value customers => customer stays for full year, independent
             # (false positive, "free"/unnecessary discount given) or correct
             pred.loc[(pred['predict_prob'] >= 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()
```
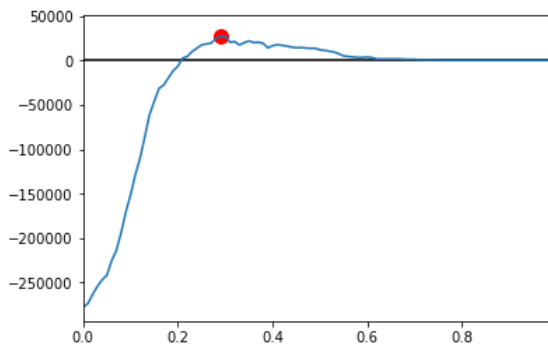
```python
# Generate a list of possible cutoffs and the corresponding overall revenue deltas
rev_deltas_high_value = pd.Series({cutoff: get_rev_delta_high_value(predictions, cutoff=cutoff) for cutoff in np.

# Generate a list of possible cutoffs and the corresponding overall revenue deltas
plot_tradeoff(rev_deltas_high_value)
```



```
Maximum benefit at cutoff 0.29 with revenue delta of $27,207.70
```

> 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.

## 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", ie seeing how well a strategy would have performed historically.

(Of course, 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 practice, actual outcomes may not be available because they are in the future.

An alternative is to optimise predicted/forecast revenue based on the probabilities which are an output from our churn model. In this case, we would replace the actual churn outcomes ( churn ) with the predicted probability 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 this path therefore usually requires the extra step of checking how well calibrated the model probabilities are, and potentially correcting for any miscalibration using Platt scaling or isotonic regression.

In [8]: 
```python
# Check our calibration
# (code copied from https://scikit-learn.org/stable/auto_examples/calibration/plot_calibration_curve.html)

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['churn'],
                                                                y_prob=predictions['predict_prob'],
                                                                n_bins=10)

ax1.plot(mean_predicted_value, fraction_of_positives, "s-", label="Our RF classifier")

ax2.hist(predictions['predict_prob'], 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()
```
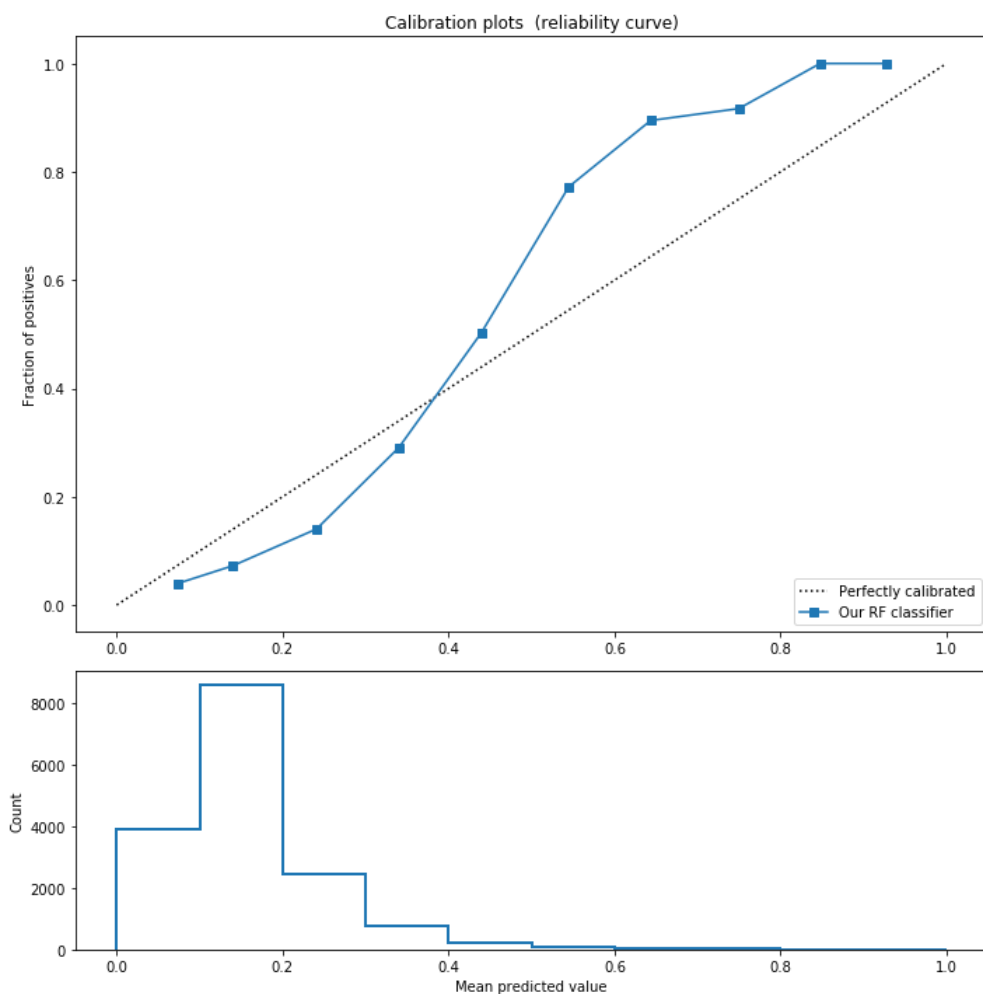
Calibration is OK, but not perfect - let's skip the calibration step here.

To use the predicted churn probability, we simply need to replace all 1/0 churn values with it in all calculations:

```python
In [9]:  # Electricity evenue 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 since we need to ask the client for more c
         # Note that we need to reverse the log10-transformation from the data cleaning step
         predictions['basecase_revenue'] = \
             np.power(10, predictions['forecast_cons_12m']) * predictions['forecast_price_energy_p1'] + \
             predictions['forecast_meter_rent_12m']

         # Taking churn into account
         predictions['basecase_revenue_after_churn'] = predictions['basecase_revenue'] * (1 - 0.919 * predictions['predict
```

```python
In [10]:  def get_rev_delta_proba(pred: pd.DataFrame, cutoff: float=0.5, discount: float=0.2) -> float:
              """
              Get the delta of revenues 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['predict_prob'] >= cutoff, '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()
```
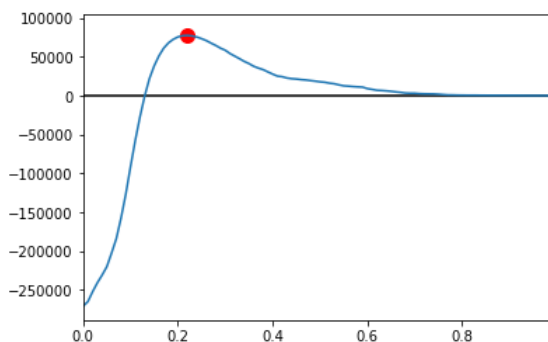
```python
In [11]:  # Generate a list of possible cutoffs and the corresponding overall revenue deltas
          rev_deltas = pd.Series({cutoff: get_rev_delta(predictions, cutoff=cutoff) for cutoff in np.arange(0, 1, 0.01)})

          plot_tradeoff(rev_deltas)
```



```
Maximum benefit at cutoff 0.22 with revenue delta of $77,390.85
```

## Next steps: How to select the discount?

In the strategy suggested by the SME division head we offer a 20% discount to all customer targeted. However, this might not be optimal either.

We assumed before that customers offered a discount will not churn. However, that may not be true in reality. The discount may not be large enough to prevent churn.

In fact, we can predict the churn probability for each customer as a function of price, margin and other factors. Therefore, we can try to find a strategy for each customer that optimises either their expected revenue or profit.

In order to go further, we'll need to try to:

- Change the level of discount offered overall
- Predict the response of customers to that discount (ie, the churn probability) based on how much that discount affects their prices, the revenue and margin.

- Take care that we've applied the discount to all affected variables. To make this easier, we might want to retrain our model using a simpler set of variables where we know that we can factor the discount correctly into the predictors.
- Find the discount level that balances customer retention vs the cost of false positives.

In fact, this could be turned into a 2D optimisation problem:

- Objective: maximise net revenue (ie including the benefits of true positives and the cost of false positives)
- Decision variables:
    - Level of discount offered, and
    - Fraction of people who are offered a discount

An even more sophisticated strategy is to find the right level of discount for *each customer* that maximises their predicted revenue or margin.

In [ ]: