The Machine Learning Canvas (v0.4)

Designed for: Priority Inbox (PI) Designed by: Louis Dorard Date: Jan. 2017 Iteration: 1

Decisions

How are predictions used to make decisions that provide the proposed value to the end-user?

Move important incoming emails to a dedicated section at the top of the inbox

ML task

Input, output to predict, type of problem.

We want to be able to answer the question "Is this email important?" before the user gets a chance to see the email

- Input: email
- Output: "Important" (Positive class) or "Regular"

-> Binary Classification

Value Propositions

What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?

Make it easier for users of an email client to identify new important emails in their inbox, by automatically detecting them and making them more visible in the inbox (this detection must happen before user sees email)

The objective is that users spend less them in their inbox and reply to important emails more quickly

Data Sources

Which raw data sources can we use (internal and external)?

- Previous email messages (as mbox files or in other type of database)
- Address book
- Calendar

Collecting Data

How do we get new data to learn from (inputs and outputs)?

- Explicit labelling: users can manually label emails as important or not, by clicking on an icon next to each email's subject
- Implicit labelling: heuristics based on user behavior after getting the email (e.g. replying fast, deleting without reading, etc.)

Making Predictions

When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?

Every time we receive an email addressed to our user, which starts a new thread (otherwise the importance is just the same as that of the thread)

We aim to rapidly deliver the email in the right section of the inbox, within a 2s period

Offline Evaluation

Methods and metrics to evaluate the system before deployment.

Use last 3 months of emails for test and 12 months before for training. We make P.I. feature available to user if...

- Cost < baseline heuristic (e.g. "if sender in address book then important"): FP costs 1, FN costs 3
- No more than 1 error per X emails

Features

Input representations extracted from raw data sources.

- Content features: subject, body, attachments, size
- Social features: based on info about sender (e.g. in address book?), previous interactions, contextual (e.g. upcoming meeting w. sender)
- Email labels (typically assigned via manual rules defined by user)

Building Models

When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?

One model per user, initially built on last 12 months of email data, that we update...

- When an error is signaled by the user via manual labelling
- Every 5' by adding new data from implicit labelling, if any

Live Evaluation and Monitoring

Methods and metrics to evaluate the system after deployment, and to quantify value creation.

Per week:

- Ratio: #errors explicitly signaled by user / #emails received
- Same w. errors seen via implicit labelling
- Average time taken to reply to important emails
- Total time spent on inbox

The Machine Learning Canvas (v0.4)

Designed for: Fake review detection Designed by: Louis Dorard Date: Jan. 2017 Iteration: 1

Decisions

How are predictions used to make decisions that provide the proposed value to the end-user?

If probability of Positive…
- > M: approve
- < m: reject

Otherwise request human decision

Thresholds m and M are chosen to maximize offline evaluation (performed right after model update)

ML task

Input, output to predict, type of problem.

"Is this review legit or fake?"

- Input: review
- Output: "legit" or "fake" (Positive class)

-> Binary classification

Note: the distribution of outputs is typically 70-30 (legit vs fake)

Value Propositions

What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?

Reject fake incoming reviews and approve legit reviews automatically.

Flag fake reviews in database to stop displaying them / using them to compute average ratings. Have ratings which are closer to the truth. Improve customer experience and satisfaction (less surprises).

Data Sources

Which raw data sources can we use (internal and external)?

- User database
- Reviews database
- Social networks
- Crowdsourcing platform (e.g. Mechanical Turk)

Collecting Data

How do we get new data to learn from (inputs and outputs)?

- Initially: active learning using crowdsourcing platform
- Internal, manual labelling
  - When explicitly requested (complaint, or model's probability in between thresholds)
  - Randomly selected reviews every day (as many as allowed for a budget of $X /day)

Making Predictions

When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?

We receive X reviews / minute on average. We can allow a delay of 1 day / review, but including 1/2 day for manual review if we're in between thresholds.

Offline Evaluation

Methods and metrics to evaluate the system before deployment.

Train model with data up until 1 wk ago. Compute total cost on last wk's data, for different values of m and M (starting at m=0 and M=1), taking into account:
- Gain of correct, automated decision = - Cost of manual decision
- Cost of FN (when review sentiment positive / negative)
- Cost of FP (smaller)

Features

Input representations extracted from raw data sources.

- Content of review: rating, text, length, # capitals…
- Other predictions: sentiment, emotion, etc.
- User: basic info, # previous bookings, # approved reviews, # rejected reviews
- Metadata (e.g. IP)
- Product being reviewed (e.g. hotel chain)
- Similarity with prev. reviews (total score)

Building Models

When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?

One model per language/ country

Somewhat adversarial setting
=> Keep on learning
=> Every week we update our models by adding all the data from last week. We allow a day for this.

Live Evaluation and Monitoring

Methods and metrics to evaluate the system after deployment, and to quantify value creation.

Every week:
- Average customer satisfaction
- # customer complaints
- # hotel complaints
- # manual reviews

The Machine Learning Canvas (v0.4)

Designed for: Real-estate deals | Designed by: Louis Dorard | Date: Jan. 2017 | Iteration: 1

Decisions

How are predictions used to make decisions that provide the proposed value to the end-user?

- Every week:
- Compute predictions for all houses currently on the market
- Filter out 50% randomly (hold out set)
- Filter out properties where asking price is higher
- Prioritize best deals first and schedule visits
- Review manually and buy at asking price or lower

ML task

Input, output to predict, type of problem.

« How much is this property worth? »
- Input: property
- Output: value

-> regression task

OR

« Is this a good deal? »
-> classification task

Value Propositions

What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?

Make better real-estate investments: compare price predictions with actual asking price of properties on the market, to find the best deals.

Data Sources

Which raw data sources can we use (internal and external)?

- Redfin
- Open data: public transports, schools, etc.
- Google Maps

Collecting Data

How do we get new data to learn from (inputs and outputs)?

Every week request Redfin data on
- New properties on the market (since previous week). Should contain property characteristics + asking price
- Sale records (initially: records for the past year). Should contain properties previously seen, but this time with actual sale price.

Making Predictions

When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?

Every week we make predictions for new properties for sale (using all property info available except asking price).

Offline Evaluation

Methods and metrics to evaluate the system before deployment.

Test on the last month of labelled data, manually review errors and compute…
- Average percentage error
- Cost: for bad deals (sale price < asking) that were seen as good deals (asking < prediction), we incur a cost of (asking - sale price) in the worst case (where we would decide to go through with investment).

Features

Input representations extracted from raw data sources.

- Property basic info
- Extracted from text description:
  - Has swimming pool
  - …
- Location:
  - Latitude, longitude
  - Address
  - Distance to closest transports and shops
  - Average rating of schools in 5 mile radius

Building Models

When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?

Only keep data up until a year ago

Update model every month (with new data available)

Live Evaluation and Monitoring

Methods and metrics to evaluate the system after deployment, and to quantify value creation.

- Investment return (should go up)
- Time spent visiting properties (should go down as we're smarter about which we want to visit)
- Sale price compared to prediction, on hold out set

The Machine Learning Canvas (v0.4)

Designed for: Customer retention | Designed by: Louis Dorard | Date: Sept. 2016 | Iteration: 1

Decisions

How are predictions used to make decisions that provide the proposed value to the end-user?

On 1st day of every month:

- Randomly filter out 50% of customers (hold-out set)
- Filter out 'no-churn'
- Sort remaining by descending (churn prob.) x (monthly revenue) and show prediction path for each
- Target as many customers as suggested by simulation

ML task

Input, output to predict, type of problem.

Predict answer to "Is this customer going to churn in the coming month?"

- Input: customer
- Output: 'churn' or 'no-churn' class ('churn' is the Positive class)
- Binary Classification

Value Propositions

What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?

Context:

- Company sells SaaS with monthly subscription
- End-user of predictive system is CRM team

We want to help them...

- Identify important clients who may churn, so appropriate action can be taken
- Reduce churn rate among high-revenue customers
- Improve success rate of retention efforts by understanding why customers may churn

Data Sources

Which raw data sources can we use (internal and external)?

- CRM tool
- Payments database
- Website analytics
- Customer support
- Emailing to customers

Collecting Data

How do we get new data to learn from (inputs and outputs)?

Every month, we see which of last month's customers churned or not, by looking through the payments database.

Associated inputs are customer "snapshots" taken last month.

Making Predictions

When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?

Every month we (re-)featurize all current customers and make predictions for them.

We do this overnight (along with building the model that powers these predictions and evaluating it).

Offline Evaluation

Methods and metrics to evaluate the system before deployment.

Before targeting customers:

- Evaluate new model's accuracy on pre-defined customer profiles
- Simulate decisions taken on last month's customers (using model learnt from customers 2 months ago). Compute ROI w. different # customers to target & hypotheses on retention success rate (is it >0?)

Features

Input representations extracted from raw data sources.

Basic customer info at time t (age, city, etc.)

Events between (t - 1 month) and t:

- Usage of product: # times logged in, functionalities used, etc.
- Cust. support interactions
- Other contextual, e.g. devices used

Building Models

When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?

Every month we create a new model from the previous month's hold-out set (or the whole set, when initializing this system).

We do this overnight (along with offline evaluation and making predictions).

Live Evaluation and Monitoring

Methods and metrics to evaluate the system after deployment, and to quantify value creation.

- Accuracy of last month's predictions on hold-out set
- Compare churn rate & lost revenue between last month's hold-out set and remaining set
- Monitor (#non-churn among targeted) / #targets
- Monitor ROI (based on diff. in lost revenue & cost of retention campaign)