1. Problem statement

1.1. Understand various dimensions of customer behavior quickly

Business challenges

- Companies want to understand various dimensions of customer behavior, at “segment-of-one” level. Such dimensions might include: **Frequency, Re-purchase behavior, Future orders**, etc.
- Fast business cycle **requires speedy implementation of ML models**, which can take 2-3 weeks to build from scratch. Slow development process could lead to major business opportunity loss.

Technical challenges

- With the availability of fast boosted tree algorithms (Catboost, Lightgbm, etc.), feature engineering has become the most important step in the ML modeling process. However, **feature engineering is a manual, often time-consuming process**, which requires a lot of analysis but doesn’t always yield the best results.
- Traditional ML models run independently and **cannot learn from each other**, which prevents generalization and induces overfitting. This is potentially problematic when there’s only a small amount of labels

Proposed Problem Statement:

- Building models that predict: **Time to next order (<1week, 1-2weeks, 2-4weeks, >=30days)**, and **Reorder rate of the next order (<10%, 10-50%, 50-90%, >90%)** (= number of re-ordered product / total number of products in the order)**.**
- Using traditional ML models (require feature engineering) as benchmark and experimenting with a sequence-based deep learning model (does not require feature engineering) to speed up development process.
- This project use data from the Instacart dataset.

1. 2. Data description

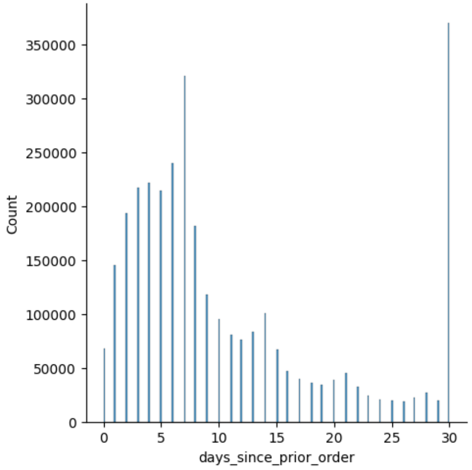
The Instacart datasets contains the following data:

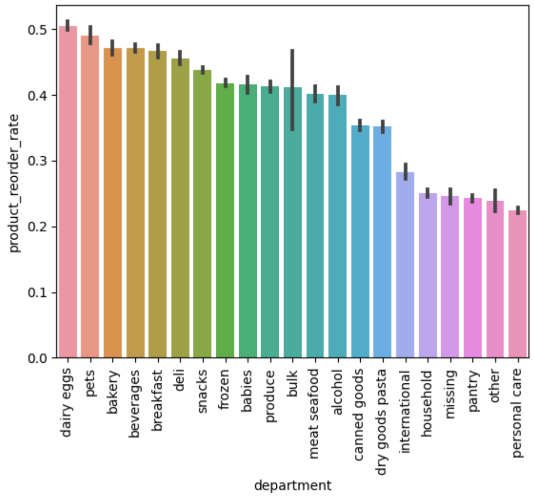
1. Historical orders for each customer, including time gaps between orders
2. Products in each order
3. Metadata of the products (department – product category level 1, aisles – product category level 2)

Data size:

- Number of customers: 206,209
- Number of transactions: 32,434,489
- Range of gaps between orders: 0 days – 30 days

A few summary of the dataset





2. Modeling

2.1. Feature-based approach: Data processing and modeling

Data preprocessing methodology

Since I only have access to raw data, I attempt to engineer features with 2 approach:

- Hypothesis based features: Calculate features that make sense business-wise, focus on historical patterns of frequency and re-order behavior (12 features)
- Build user-product interaction matrix, then apply PCA to extract features with the highest variance. (42 features)

⇒ Result in a dataset of 54 features in total

Train-test split: Set aside 10% as test data, apply cross-validation to the 90% training data

Decision made:

- User-product interaction matrix was too large, couldn't fit in memory to perform PCA => Decided to use the user-department interaction matrix and the user-aisle interaction matrix instead.
- Re-formulated the problem of Time to next purchase from regression problem to classification problem due to data distribution (the data was capped at 30 days)

Modeling methodology

5-fold cross validation to tune hyperparameters for the most popular models:

- Logistic regression
- Random Forest
- Neural network (multi-layer perceptron)
- Boosted tree

Evaluation matrix: Macro F1-score as main matrix, also keep track of ROC curve's AUC

Decision made:

- Used RandomSearch instead of GridSearch because not all hyperparameters are equal
- Dropped a few models due to slow training time or slow inference time: Xgboost, SVM, KNN
- Used LightGBM instead of Xgboost as representation of boosted tree for its significantly faster training time
- Chose Macro F1-score (instead of micro) due to unbalanced data

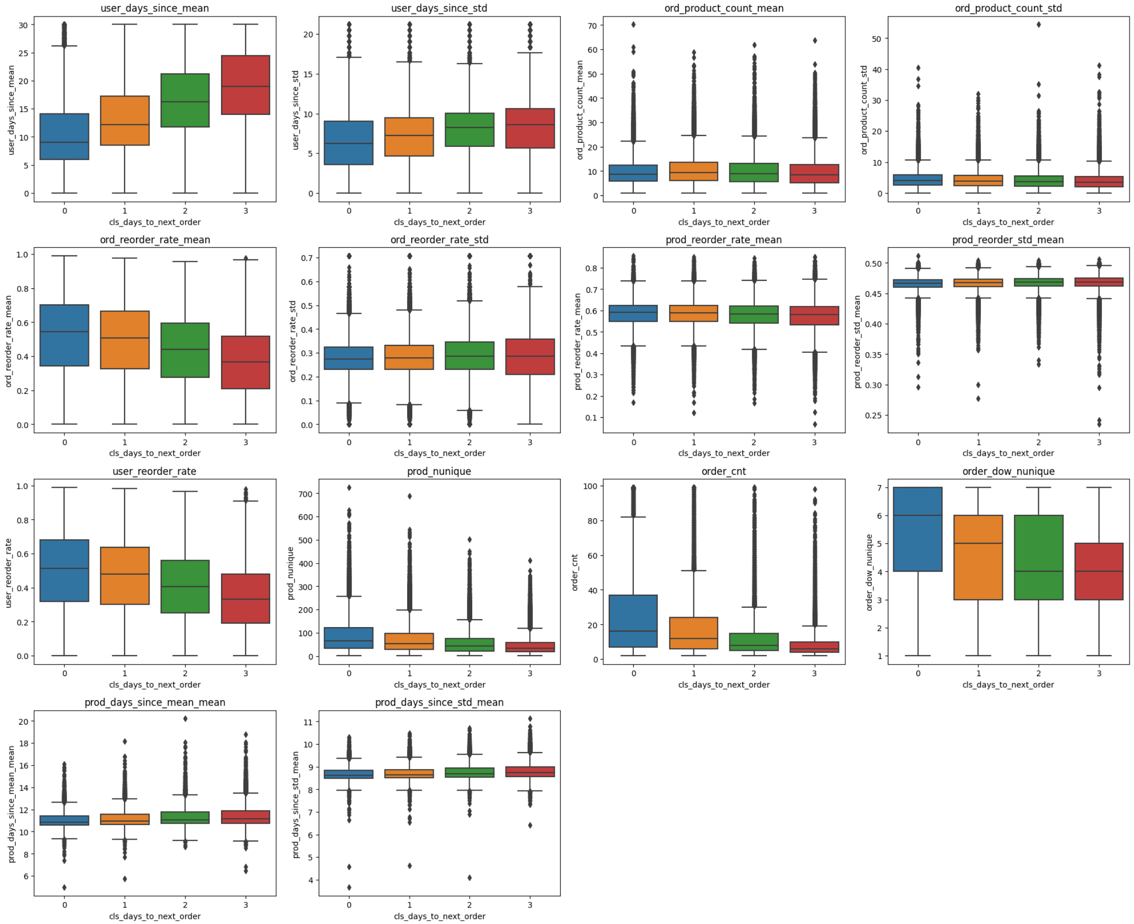
2.1. Feature-based approach: Feature engineering results (1/2)

Hypothesis-based features and correlation with task 1- **Time to next order**

Overall, we can see that user_* features and ord_* features offers strong signals.

Meanwhile prod_* features don't seem to differentiate much between customers who have short time to next order and customers who have long time to next order.

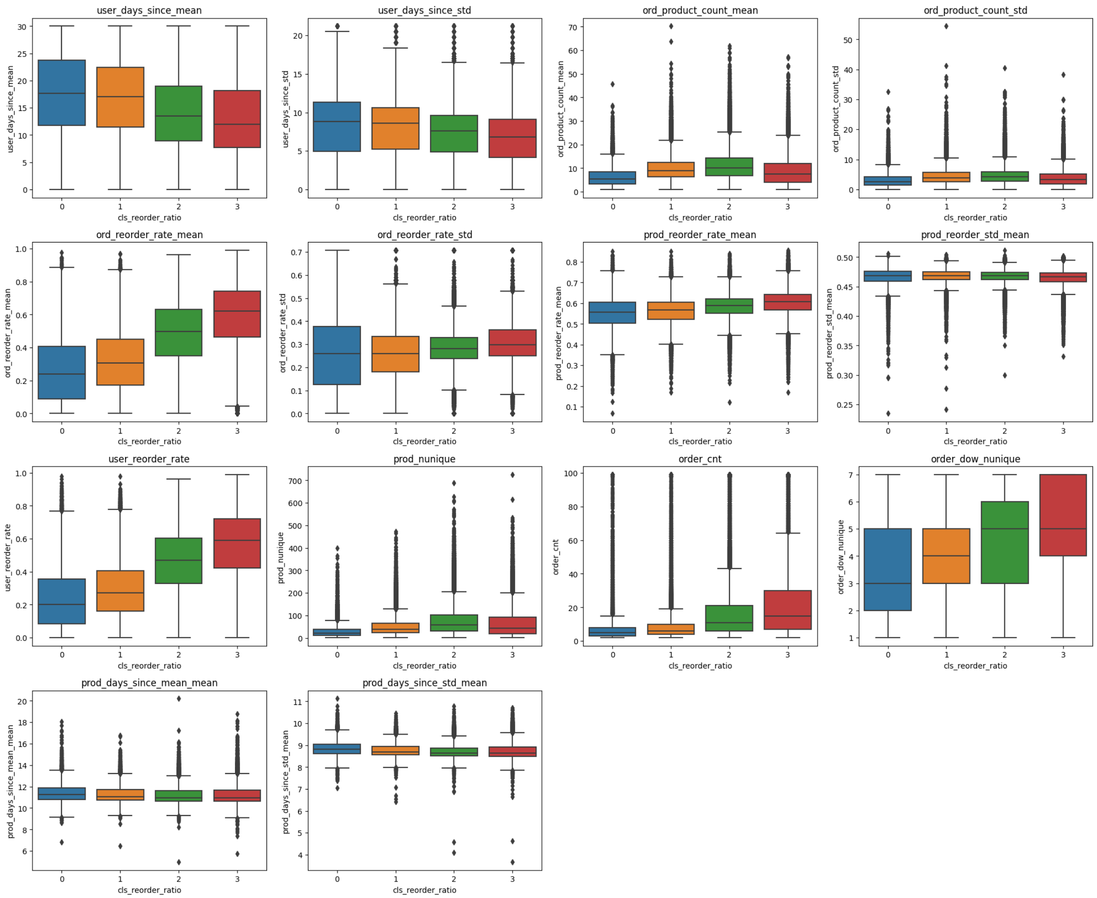
List of 12 features: ['user_days_since_mean', 'user_days_since_std', 'ord_product_count_mean', 'ord_product_count_std', 'ord_reorder_rate_mean', 'ord_reorder_rate_std', 'prod_reorder_rate_mean', 'prod_reorder_std_mean', 'user_reorder_rate', 'prod_nunique', 'order_cnt', 'order_dow_nunique', 'prod_days_since_mean_mean', 'prod_days_since_std_mean']



2.1. Feature-based approach: Feature engineering results (2/2)

Hypothesis-based features and correlation with task 2 - **Reorder ratio**

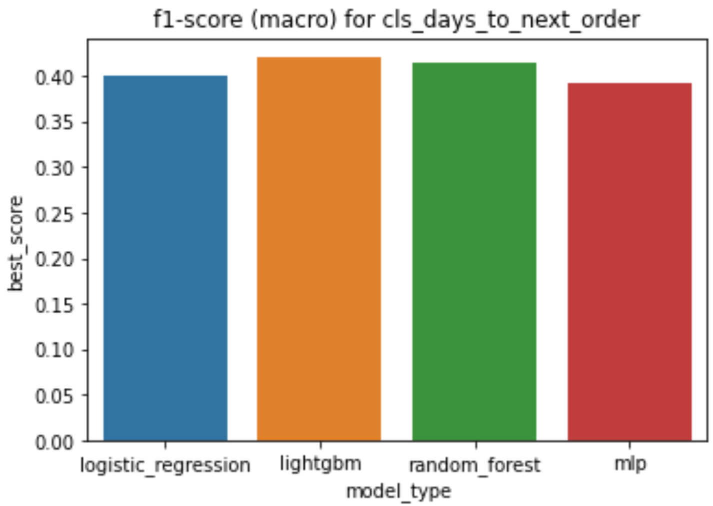
Similar to task 1, we can see that user_* features and ord_* features seem to differentiate well between customers who have high proportion of reorder products in the next order and customers who have low proportion.



2.1. Feature-based approach: Hyperparameter tuning results

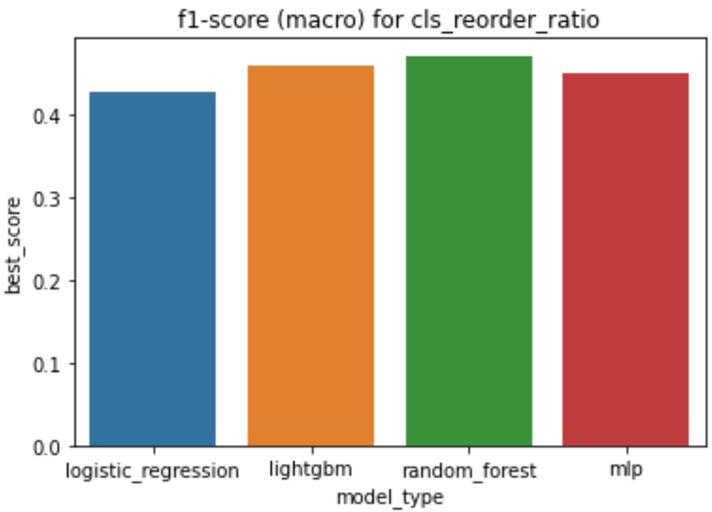
Task 1 – Time to next order

LightGBM performs the best for this task, achieving a 5-fold cross-validated F1-score of 0.42, while MLP comes last at 0.39



Task 2 – Reorder rate

RandomForest performs the best for task 2, achieving a cross-validated F1-score of 0.47, while LogisticRegression has the lowest performance at 0.42 macro F1-score



2.1. Feature-based approach: Performance on test set – Task 1

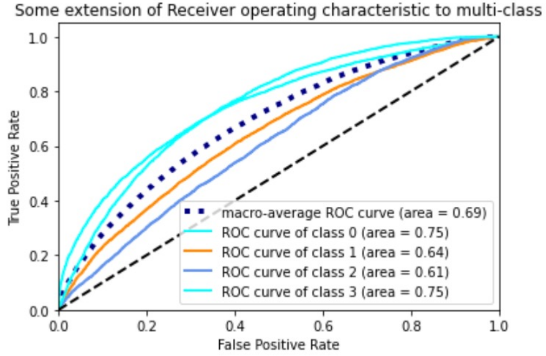
Performance metrics on the test set are consistent with cross-validated performance => No overfitting

LogisticsRegression performance on test set

Evaluating model

	precision	recall	f1-score	support
0	0.43	0.55	0.48	4587
1	0.41	0.28	0.33	5762
2	0.29	0.20	0.24	4380
3	0.47	0.62	0.54	5892
accuracy			0.42	20621
macro avg	0.40	0.41	0.40	20621
weighted avg	0.41	0.42	0.40	20621

```
0: 0.7488951630716927
1: 0.6447946226170104
2: 0.609174536849211
3: 0.7490585281946285
```

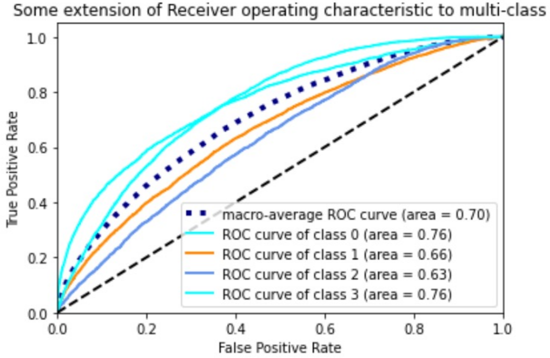


LightGBM performance on test set

Evaluating model

	precision	recall	f1-score	support
0	0.49	0.51	0.50	4587
1	0.43	0.32	0.36	5762
2	0.30	0.26	0.27	4380
3	0.48	0.62	0.54	5892
accuracy			0.44	20621
macro avg	0.42	0.43	0.42	20621
weighted avg	0.43	0.44	0.43	20621

```
0: 0.7649349013877449
1: 0.6599115686060562
2: 0.6300073746499291
3: 0.7589258795831473
```



2.1. Feature-based approach: Performance on test set – Task 1

Performance metrics on the test set are consistent with cross-validated performance => No overfitting

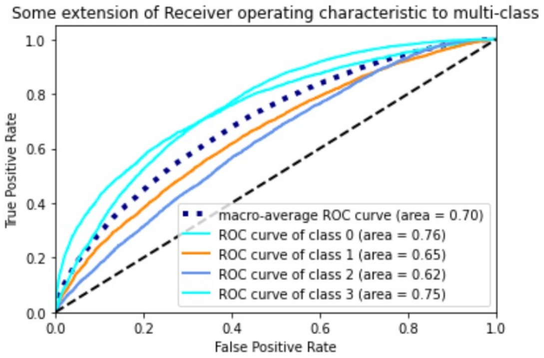
RandomForest performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.50      0.48      0.49      4587
           1       0.41      0.33      0.37      5762
           2       0.29      0.23      0.25      4380
           3       0.46      0.65      0.54      5892

    accuracy                           0.43     20621
   macro avg       0.42      0.42      0.41     20621
weighted avg       0.42      0.43      0.42     20621

0: 0.7573070077622005
1: 0.6537248819920792
2: 0.6230309501939817
3: 0.7530673885201005
```



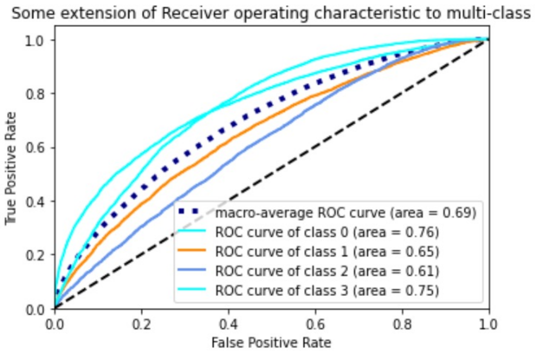
NeuralNetwork performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.52      0.43      0.47      4587
           1       0.41      0.39      0.40      5762
           2       0.34      0.07      0.11      4380
           3       0.43      0.77      0.56      5892

    accuracy                           0.44     20621
   macro avg       0.43      0.42      0.39     20621
weighted avg       0.43      0.44      0.40     20621

0: 0.7574714310899019
1: 0.6521413166210603
2: 0.6108430127370861
3: 0.7515082515675717
```



2.1. Feature-based approach: Performance on test set – Task 2

Performance metrics on the test set are consistent with cross-validated performance => No overfitting

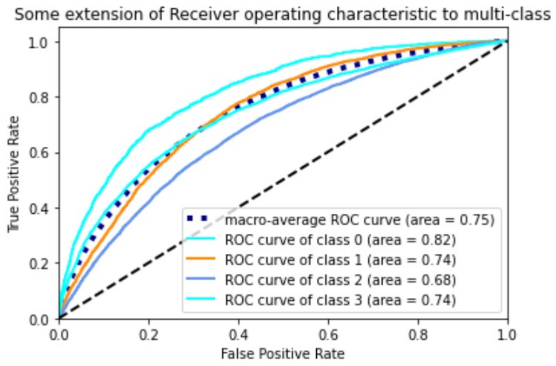
LogisticsRegression performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.22      0.61      0.32      1554
           1       0.47      0.43      0.45      5380
           2       0.63      0.38      0.47      9041
           3       0.43      0.55      0.48      4646

    accuracy                           0.45     20621
   macro avg       0.44      0.49      0.43     20621
weighted avg       0.51      0.45      0.46     20621

0: 0.8167574965445632
1: 0.7441378408709242
2: 0.6809540265522311
3: 0.7394854745731769
```



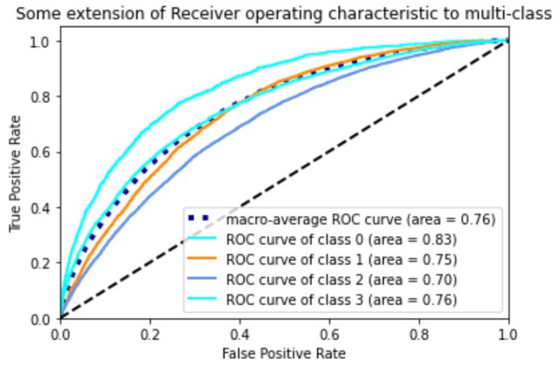
LightGBM performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.26      0.54      0.35      1554
           1       0.46      0.49      0.47      5380
           2       0.62      0.44      0.51      9041
           3       0.46      0.53      0.49      4646

    accuracy                           0.48     20621
   macro avg       0.45      0.50      0.46     20621
weighted avg       0.52      0.48      0.49     20621

0: 0.8292921749417266
1: 0.7456848444166817
2: 0.6957617944275731
3: 0.7585339770964237
```



2.1. Feature-based approach: Performance on test set – Task 2

Performance metrics on the test set are consistent with cross-validated performance => No overfitting

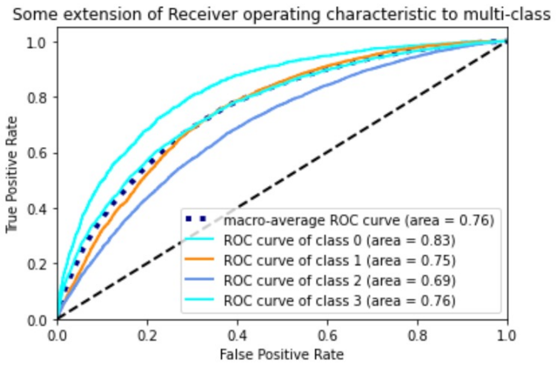
RandomForest performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.34      0.35      0.34      1554
           1       0.47      0.58      0.52      5380
           2       0.60      0.50      0.55      9041
           3       0.48      0.51      0.49      4646

    accuracy                           0.51     20621
   macro avg       0.47      0.48      0.48     20621
weighted avg       0.52      0.51      0.51     20621

0: 0.8253191904264439
1: 0.7516220554564592
2: 0.6916736536434768
3: 0.761338415531694
```



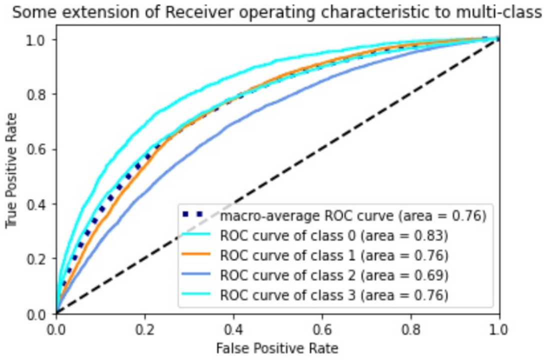
NeuralNetwork performance on test set

```
Evaluating model
              precision    recall  f1-score   support

           0       0.41      0.21      0.27      1554
           1       0.49      0.48      0.49      5380
           2       0.56      0.68      0.62      9041
           3       0.53      0.41      0.46      4646

    accuracy                           0.53     20621
   macro avg       0.50      0.45      0.46     20621
weighted avg       0.53      0.53      0.52     20621

0: 0.8253651571687971
1: 0.7554786065467609
2: 0.6936198156202247
3: 0.7646930706542792
```



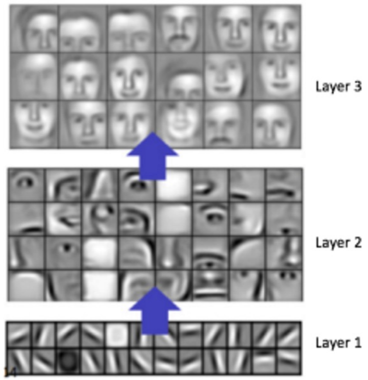
2.2. Sequence-based approach: Foundational concepts

Representation learning (with neural networks)

What it does
Representation learning or feature learning automatically discovers the representations needed for the learning task from raw data.

Problem it solves
By eliminating the feature extraction/feature engineering step, representation learning helps:
- Reduce information loss due to feature extraction
- Improve time-to-production of ML models
- Produce good results when domain knowledge is not available

Basic principle
Multi-layer neural network learn the representation by learning fundamental concepts in the shallower layers and high-level concepts in deeper layers



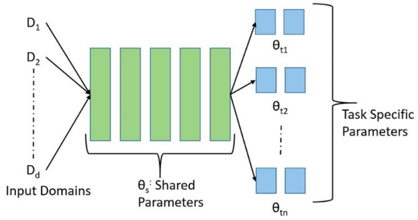
Multi-task learning

What it does
Multi-task learning solves multiple tasks at the same time, while exploiting commonalities and differences across tasks.

Problem it solves:
- Solve many different problems at the same time
- Reduce or even eliminate overfitting
- Improve the model's generalization a lot by not just considering one particular aspect

Basic principle
Theories on why multi-task learning works includes:
- Transfer knowledge between tasks by using some form of shared representation => Improve learning ability
- Introduce a better form of regularization compare to L1, L2 => reduce overfitting

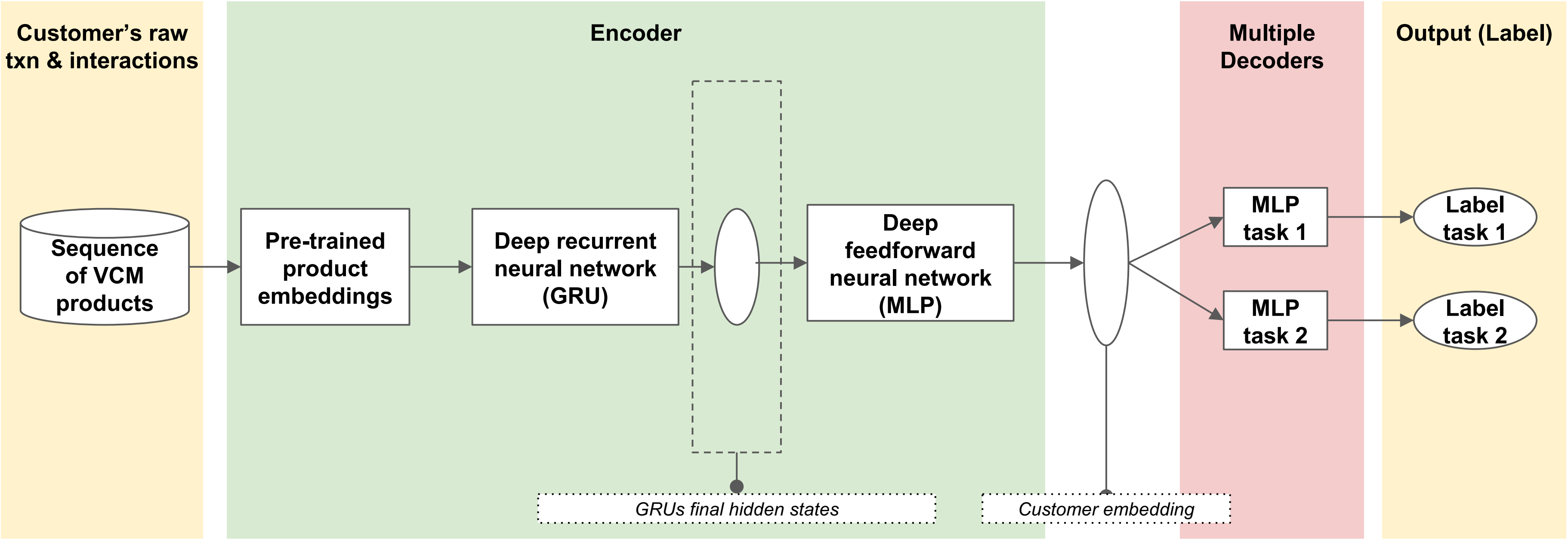
In general, multi-task learning is deemed to resemble human learning better compared to single-task learning



2.2. Sequence-based approach: Network architecture

Inspired by Sutskever (2014) which describe the proposed neural machine translation architecture as a model that wants to work, the model uses *Sequence encoding* to encode customer's interactions and *Multilayer perceptron (MLP)* to decode different tasks

Note: Due to time constraints, no model tuning was performed for the sequence approach.



2.2. Sequence-based approach: Performance on test set

Model’s performance for Time to next order

	precision	recall	f1-score	support
0	0.43	0.44	0.43	4484
1	0.27	0.36	0.31	4321
2	0.21	0.28	0.24	3388
3	0.66	0.46	0.54	8428
accuracy			0.40	20621
macro avg	0.39	0.38	0.38	20621
weighted avg	0.45	0.40	0.42	20621

For task 1 – Time to next order, F1-score is comparable with baseline, only slightly lower (0.38 compared to baseline models ranging between 0.39 and 0.42).

Model’s performance for Reorder rate

	precision	recall	f1-score	support
0	0.57	0.21	0.31	4185
1	0.41	0.42	0.41	5290
2	0.36	0.58	0.45	5699
3	0.48	0.41	0.44	5447
accuracy			0.42	20621
macro avg	0.46	0.40	0.40	20621
weighted avg	0.45	0.42	0.41	20621

For task 2 – Reorder rate, the gap is more significant (Sequence-based model’s F1-score is 0.40 while baseline models range between 0.43 and 0.48).

- Sequence-based models perform slightly worse than well-tuned feature-based models. This is promising for an out-of-the-box deep-learning model.
- With more efforts put into hyperparameter tuning, the performance might be improved to be on-par with feature-based approach.
- Regarding multi-task learning aspect of the model, with only 2 tasks incorporated, multi-task learning’s advantages were not expressed significantly.

3. Summary and discussion

3.1. Summary

Pros & Cons of the model

Pros and cons of Sequence-based approach compared to feature-based approach:

Pros:

- Save weeks spent on feature engineering with built-in representation learning
- Potential to incorporate predictor data from different sources with a plug-and-play mechanism of encoders and decoders
- Produce **product embeddings** and **user embeddings,** which can be re-used easily by other models

Cons:

- Is a black-box model, cannot easily be explained like tree-based models or linear models
- Is a big model and therefore take a lot of resources (time, computational resources) to train and tune

Main contribution of this research

The main contributions of this research are:

1. Combining **representation learning and multi-task learning** into a single model

2. Proposing a **encoder architecture** for some of the most unstructured, hard-to-mine data (products sequence)

3. Demonstrating through experimentation that the model **produces acceptable performance,** not much worse compared to baseline models

3.2. Potential future work

The limits of the current models

- Training time is too slow on a single GPU. Take 30 minutes to train a single model, 50x slower compared to the slowest of feature-based approaches (RandomForest)
- Data input only consider products bought as sequences and does not including the monetary values or product metadata.

Unexplored territories:

- Currently use Hard Parameter Sharing (HPS) between encoders, there are other multi-task learning architectures to be explored
- Currently use Equal weight to calculate loss function, other weighting strategies that can take into account the importance of each task
- Other training scheduler can make training faster, focusing on rate of improvement for each task
- Task grouping might help improve performance by having different level of parameter sharing

Future work

1. Reduce training time by experimenting with training schedulers, weighting strategies, and hardware acceleration
2. Improve performance with other multi-task learning architectures
3. Develop encoders to efficiently capture more user information and actions. Some ideas could be:
    a. Monetary value and product metadata
    b. Graph embeddings
    c. Convolutional neural network
4. Apply to real problems

Thank you