Task 1

Implement a handful of classification methods and describe how to assess how well each method works? How might your assessment methodology differ if you had labeled data that assigned each transaction to a category? Implement your assessment methodology and show us how you assess the different classification methods.

Background

The objective of a classification model in this context is to create useful abstraction layers (categories/subscriptions/services) from the raw data, which can benefit users. Without explicit labels, we need a semi-supervised or unsupervised modeling approach that can cluster the text data into meaningful categories or labels. Latent Dirichlet Allocation (LDA) or a pre-trained transformer model from Hugging Face for Named Entity Recognition (NER) could work as a semi-supervised approach. However, this would require a human to verify that the clusters or topics are meaningful. Alternatively, a BERTopic model could be used with guided topic suggestions that nudge the model to converge towards the input seed topics. BERTopic models leverage hugging face transformers and c-TF-IDF to create dense clusters allowing for easily interpretable topics while keeping important words in the topic descriptions.

If explicit labels for the transactions existed, then a direct supervised learning approach can be taken for classifying transaction descriptions into a set of target class labels. For supervised text classification, we can use a fine tuned transformer, SVMs, or even Convolutional Neural Networks (CNNs). Although CNNs are not as common for NLP tasks, they work relatively well for text classification and sentiment analysis where explicit labels are provided. In the supervised learning approach with explicit labels, weighted precision, which implicitly accounts for class imbalance, could be the primary metric of interest used for model performance comparisons and model selection.

Without explicit class or category labels, supervised approaches aren't feasible. Comparing model performance for a semi-supervised approach like LDA or a pretrained transformer would need to use a purity score, topic coherence score, or human evaluation to infer the usefulness of topic categorizations. An approach that tests multiple BERTopic models seems like the best solution to this problem. Using a BERTopic model with seed topics like "subscription", "credit card", or other expense categories could be meaningful to users that are trying to understand their monthly expenses and help them make informed financial decisions.

Models Tested

Four BERTopic models were tested. Their performance was evaluated using an average purity score with 5-fold cross validation.

1. **BERTopic.UMAP.01** - baseline BERTopic model with UMAP dimensionality reduction and a 0.01 minimum distance parameter.
2. **BERTopic.UMAP.1** - same as model #1 but with a 0.1 min distance parameter.
3. **BERTopic.MPNET.01** - baseline BERTopic UMAP model but the original MiniLM "all-MiniLM-L6-v2" has been swapped for the MPNET dimensionality reduction and an "all-mpnet-base-v2" embedding model, which should have better performance. It uses a 0.01 minimum distance parameter for dimensionality reduction.
4. **BERTopic.MPNET.1** - Same as model #3 but with a 0.1 minimum distance parameter.

The training and validation performance for each model is in the table below. After testing, the best model was trained using the full training set and evaluated on a 10% hold out test set.

Model	Training Purity Score	Validation Purity Score
BERTopic.UMAP.01	0.7254	0.5141
BERTopic.UMAP.1	0.7357	0.5503
BERTopic.MPNET.01	0.7298	0.5310
BERTopic.MPNET.1	0.7401	0.5709

The best model with the highest purity score was the BERTopic.MPNET.1 with an average cross validation purity score of 0.57. The way the model grouped transactions resulted in two topics that seem to capture subscriptions. Namely topic 1 and 4.

Topic 1: `[('membership', 0.4231301426050734), ('annual', 0.4231301426050734), ('fee', 0.35260845217089454), ('planet', 0.2825035479513955), ('fit', 0.2670137510465075), ('fees', 0.23418833275149298), ('club', 0.22729193739412423), ('ach' …]`

Topic 4: `[('instacart', 0.37682105282649553), ('8882467822', 0.17492091237937776), ('ca', 0.16793961789161077), ('httpsinstacar', 0.16458372315125078), ('card', 0.10672800498319503), ('subscription', 0.0951936688015706) …`

Having a data scientist review the output topics and merge them as necessary before deployment might be necessary before deployment. This model still requires an explicit hashmap or taxonomy that aligns with the product use cases.

Product Use Cases

1. Identification of subscription expenses. For example, transactions that are classified as “subscription” expenses could be used to manage monthly subscription costs and help users cancel unnecessary subscriptions to reduce their monthly expenses.
2. Categorizing “credit card” expenses can help users understand their total credit card spend over time and reconcile the spend with their predefined budget goals. If data exists on the total credit card balances for users, then a debt repayment feature could help users optimize a debt repayment strategy by minimizing total interest charges over time or offering debt negotiation services.
3. General expense categorization e.g. dining, entertainment, bills, rent, insurance. Matching transactions into different expense categories can help users better understand their monthly spending habits and align these expenses with their budget goals.

Scaling Production

- Calculating the probabilities for the topics is computationally expensive and should be turned off when scaling up into production.
- The current model does not scale well due to the small sample of data used for training and the fact that a very short seed topic list was used for guiding the model training.
- Adding new groups can be done automatically by retraining models on a fixed cadence. Ideally, we would use a two step procedure that retrains a BERTopic model with new data periodically as part of a batch job.

1. Using a pre-trained model for NER to inform the creation of a more expansive seed topic list would make the model more scalable and robust to new entities and topics.
2. Use an online learning approach that incrementally updates the topic model as new data is batched with the update seed topic list. Online BERTopic modeling. This would require a data scientist to review the resulting topic list and update it periodically.

How to deal with new entities or topics e.g. a cool new competitor to Netflix arises and lots of people start paying for it?

- Creating a rules based alert threshold would help with post deployment monitoring of new categories, subscriptions, or services. For example, if a new competitor to Netflix arises as a transaction within the transaction history of many users, an automated alert could notify the team of the new transaction entity. Automated identification is possible with a pre-trained NER model. If an NER model is used to generate entities for the seed topic list, a data scientist could set alerts and approve new entities before they are added into the seed topic list used by the BERTopic model training pipeline.
- If the model fails to properly classify a new subscription entity or topic, a user feedback mechanism would generate data that can be used to infer performance degradation in production and generate an alert.

Post Deployment

Having a product feature that allows users to reclassify transactions that have been misclassified would allow for a “human in the loop” feedback mechanism. These user provided data tags can be used for retraining the BERTopic model or training other models with explicit class labels. Additionally, the user action of recategorizing a transaction is a signal that could be used to infer production model classification performance over time. This rate of user transaction recategorization can be used for post deployment monitoring of the model. For example, when the model incorrectly groups a transaction, the user can select the transaction and label it as something else e.g. investment, dining, or subscription. Also, a user provided tag like “subscription” could be added to the training data and used to update the model. Hypothetically, this could be done in an automated way as part of the training pipeline.

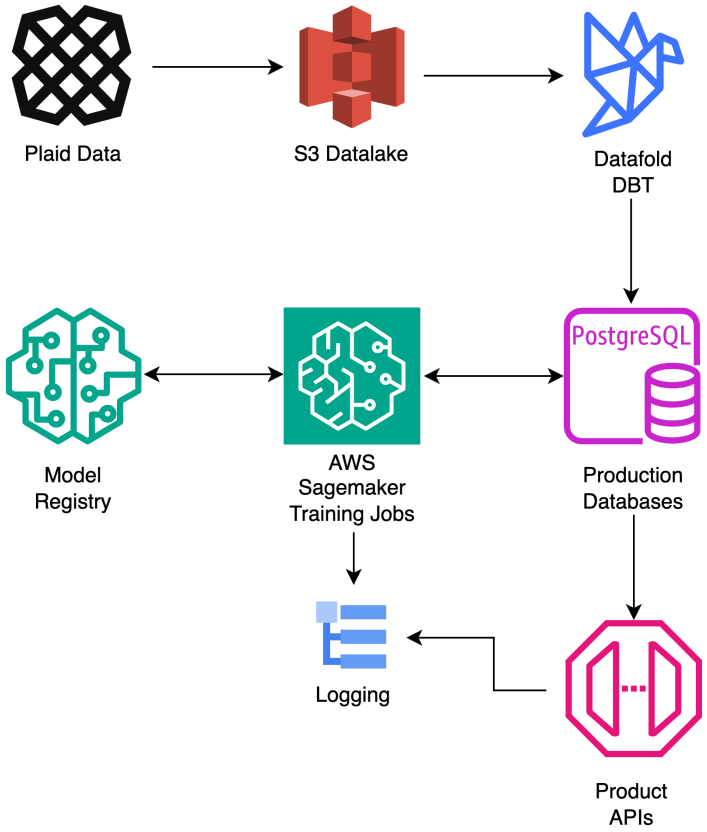
The BERTopic model outputs a “-1” value for unclassified text. This value can be converted into a percentage of unclassified transactions over time, which can be used to infer the coverage of the model. If the model performance breaks down, then retraining should be considered.

Future Considerations

1. Run a full test with more data that optimizes the hyperparameters of a handful of BERTopic models. Due to computational constraints, the models tested for this assessment did not benefit from a comprehensive hyperparameter optimization procedure.
2. Training models with a more extensive list of seed topics that align with the product goals would result in more meaningful topics or categories. The semi-supervised approach could be improved by using a pre-trained model for Named Entity Recognition that updates the seed topic list of an online BERTopic model.
3. Integrating user feedback tags like “subscription” into the training data pipeline could improve the classification performance of the model over time.

Task 2

The proposed high level infrastructure design below assumes Rocket Money is using AWS. If cost is a concern, then MLFlow could be used instead of AWS SageMaker for model version management. However, this would require the creation of a separate PostgreSQL database for MLflow. Also, this design is production only and ignores the existence of any staging environments.



- Plaid sources transaction data into an S3 data lake.
- Datafold is being used for batch DBT model management.
- Training model pipelines can be managed within AWS SageMaker.
- Model versions are stored in the Model Registry.
- AWS Cloud Watch could be used for all logging of AWS hosted microservices.
- Using a batch job:
    - Load the data from a production database.
    - Assign transactions to categories using a trained BERTopic model.
    - Categorized transaction data can be written to a production database, which allows the product APIs to quickly query the necessary data for user devices.