Tide Case Study

Business Problem :

'Receipt Importer' is a new feature that we have launched. This tool aims to reduce the time customers spend doing the admin stuff. We are tasked to improve the performance of this tool.

Data Science Problem :

Customers take the image of the receipt and upload it. Use Image processing to extract data from the scanned receipt image. Match the scanned transaction id against the same data from the tide transaction

Solution A: Heuristics Based Approach

Use Fuzzy logic to match the transaction id, date from the extracted feature vector of the scanned match and combine it with the data in the tide transaction table.

Challenge: Data extraction from the receipt image is not always perfect. Hence relying entirely on the extracted data is not a good idea. This leads us to the next solution

Solution B: Machine Learning-Based Approach

Build a model to learn which matching features are the most successful For each transaction, get the likelihood of that being the correct transaction. Next, we will sort the possible transactions for a given receipt by that likelihood

Approach for solution B in detail :

Data

For each receipt_id & company_id we are provided with matched_transaction_id & feature_transaction_id to match for

Following features are available in the dataset:

'DateMappingMatch', 'AmountMappingMatch', 'TimeMappingMatch', 'DescriptionMatch', 'ShortNameMatch', 'DifferentPredictedTime', 'DifferentPredictedDate', 'PredictedNameMatch', 'PredictedAmountMatch', 'PredictedTimeCloseMatch'

Target Label

I have approached this as a Supervised Machine Learning problem and for that, I need to have a target label.
However, the target label is not available, so I had to create one using the matching logic on matched_transaction_id & feature_transaction_id

Success Criteria of Model

Given the nature of the problem, I have tracked ROC AUC Score as evaluation metric In this situation my primary focus is on reducing the number of False Negatives (that is to avoid a true match of a receipt being not learned by model) Or to say I want to avoid Type II error I am still okay to have a slightly more False Positive because it will not have damage to the customer experience as long as the current receipt has the highest probability of matching

Feature Engineering

The polarity of 2 Features is turned opposite under the hypothesis that these features are not aiding in the positive direction

NegDifferentPredictedTime is -1 * DifferentPredictedTime

NegDifferentPredictedDate is -1 * DifferentPredictedDate

This is done because when subsequent aggregate features are calculated instead of adding the values of these 2 features they will be subtracted

A lot of interaction features were created in the combination of 2. Ex,

DateMappingMatch & PredictedNameMatch were added to create

DateMappingMatch_PredictedNameMatch

A lot of interaction features were created in the combination of 3. Ex, DescriptionMatch ,

PredictedNameMatch , ShortNameMatch were added to create

DescriptionMatch_PredictedNameMatch_ShortNameMatch

Model Development

Baseline models were created using 24 different algorithms

Choice of Model — LightGBM

By comparing the performance from various models and given the nature of feature space, I decided to use LightGBM because it uses EFB (Exclusive Feature Bundling) to bundle the sparse (mostly 0) mutually exclusive features. Like categorical variables that are one-hot encoded. It is a type of automatic feature selection.

Cutoff Threshold Value Evaluation

Threshold value of 0.13 is found to be most suitable for maximizing the ROC AUC Score as well as keeping the value on False Negative in-check

Using the model

Files provided in the package

1. requirements.txt
2. model_training.ipynb
3. lgb_model2.joblib
4. model_predictions.py
5. config.py

To generate predictions

Success of the code will be determined by the column pred_prob_rank

Rank = 0, means no match found

Rank = 1, means the highest probability of the receipt to match

Subsequently, ranks are provided till 5 Ranks. Higher the rank lower the probability from the model

Following code will generate predictions for the input data

```
# load libraries
import pandas as pd
from model_prediction import generate_prediction
from config import THRESHOLD, MAX_RANKS, FEATURES

# load input data
input_df = pd.read_csv('file.csv', sep=':')

# use prediction function to generate output
preds = generate_prediction(data = input_df, threshold=THRESHOLD,
max_ranks=MAX_RANKS, features=FEATURES)
```