Week 6 — Develop First modeling approach

Maria Alice Fagundes Vieira

Aritra Ray

- **The Model Approach**

This week the modeling approach we used was collaborative filtering, which is a model frequently used on product recommendation systems. This method works well for recommendation tasks by identifying patterns in how users interact with items and recommending products that similar users have liked or purchased. In other words, it takes advantage of customer interactions with products and similar customer's behavior, to help predict the possibility that a customer will buy a product. This approach is effective because it doesn't require detailed product features, and instead relies on the collective preferences of users, making it scalable and adaptable to new data.

- **Complexity of the modeling approach**

The complexity of a collaborative filtering model primarily comes from its reliance on user-item interactions and the need to capture underlying patterns in large datasets. Traditional user-based or item-based collaborative filtering methods, such as k-nearest neighbors, can become computationally expensive as the number of users and items increases, leading to higher memory usage and slower performance during both training and inference. Model-based approaches further add complexity due to the need for optimization algorithms to minimize loss functions. Additionally, handling sparsity in the user-item interaction matrix, ensuring scalability for real-time recommendations, and addressing issues like cold-start for new users or items increase the overall complexity of collaborative filtering systems.

The model complexity in our code is primarily determined by the hyperparameter variations used for the logistic regression models. The code trains multiple logistic regression models for each product, using different sets of hyperparameters. For our model we calculated ROC AUC using probabilities (predict_proba() method), which is more appropriate for this metric since ROC AUC works with predicted probabilities for the positive class and not binary predictions. We first ran the model with just the predict() method and the ROC AUC scores were considerably worse than when the probability was used. F1 Score and confusion matrix were calculated using the binary predictions (predict() method), which is the correct approach for these metrics.

- **Hyperparameters Evaluated**

Using various hyper-parameter adjustments we assessed three iterations of the collaborative filtering model in our work.

```
# Define hyperparameter variations
hyperparameter_variations = [
    {'C': 0.01, 'solver': 'liblinear', 'max_iter': 100},
    {'C': 1, 'solver': 'lbfgs', 'max_iter': 200},
    {'C': 10, 'solver': 'liblinear', 'max_iter': 300},
]
```

The complexity of the hyperparameters varies across three main dimensions:

1. Regularization strength (C parameter): Lower values of C increase regularization, reducing model complexity, while higher values allow for more complex decision boundaries.
2. Solver algorithm: Different solvers (e.g., 'liblinear', 'lbfgs') can affect how the model optimizes its parameters, potentially leading to different levels of complexity.
3. Maximum iterations (max_iter): This parameter controls how long the model trains, potentially affecting its ability to fit more complex patterns in the data.

By testing different combinations of these hyperparameters, the code explores various levels of model complexity to find the best balance between fitting the training data and generalizing to new data for each product. These factors assist the model in identifying underlying client's preferences to finally suggest a product to buy.

A model may produce recommendations that are less individualized if there are insufficient latent factors to account for the diversity of customer preferences. However an overabundance of factors may lead to overfitting a situation in which the model performs poorly on new clients because it is overly tailored to the training set.

- **Model Performance Metrics**

This week we used ROC AUC as one of the metrics to evaluate our collaborative filtering model due to its threshold independence and ability to handle imbalanced datasets, which are common in recommendation systems, as the distribution of interactions may be skewed (e.g., a client might only buy a small fraction of the available products). It provides a comprehensive view of model performance by plotting True Positive Rate against False Positive Rate across all classification thresholds. The AUC score is interpretable as the probability of ranking a positive

example higher than a negative one, making it useful for comparing different models: A higher AUC value indicates better performance, which can guide model selection or tuning. Additionally, in recommendation systems, accuracy alone can be misleading due to class imbalance. ROC AUC provides a more nuanced understanding of model performance, helping to avoid potential pitfalls of using accuracy in such scenarios.

While ROC AUC is robust and informative, it's best used alongside other metrics to fully assess a recommender system's performance, so we also used F1 score and a Confusion Matrix to evaluate the model as a whole. The F1 score is the harmonic mean of precision and recall, providing a single score that balances both metrics. The confusion matrix is a more generic metric that provides us a tabular representation of actual vs. predicted values, providing insights into true positives, true negatives, false positives, and false negatives. We believe that all three together will give us a holistic view of the model performance

- **Training and Validation Metrics Across All 3 Variations**

ROC AUC Scores:

Variation 1 shows the lowest average ROC AUC scores for both training (0.8276) and validation (0.8365) datasets. There is a significant improvement on variations 2 and 3 from variation 1, with very similar performance to each other. Variation 2 has slightly higher average ROC AUC scores for both training (0.8873) and validation (0.9075) compared to Variation 3 (0.8880 for training and 0.9061 for validation).

F1 Scores:

Variation 1 has the lowest average F1 scores for both training (0.0755) and validation (0.0858) datasets. Variations 2 and 3 show substantial improvements in F1 scores compared to Variation 1. Variation 2 and 3 have the same average F1 scores for training (0.1107) and Variation 2 has a slightly higher average for validation (0.1207) compared to Variation 3 (0.12).

Overall, the improvement from training to validation is most pronounced in Variation 1, while Variations 2 and 3 show more consistent performance between training and validation sets. The improvement from Variation 1 to Variations 2 and 3 suggests that the more complex models (with higher C values and different solvers) are better suited for this dataset. Additionally, the similarity between Variations 2 and 3 indicates that increasing the C value from 1 to 10 doesn't significantly impact performance.

Variation	Dataset	AvgROC AUC	AvgF1 Score
Variation 1	train	0.827581	0.075498
Variation 1	val	0.836491	0.085847
Variation 2	train	0.887266	0.110663
Variation 2	val	0.907466	0.120787
Variation 3	train	0.888049	0.110734
Variation 3	val	0.906114	0.119955

- **Identify the Best Model For the Week**

Variation 2 is slightly favored due to its marginally higher ROC AUC score, which is often considered a more robust metric for imbalanced datasets. So we defined variation 2 to be the best model for the week since it has a higher average ROC AUC score on the validation dataset. Both Variations 2 and 3 have the same average F1 scores for the validation dataset.

One interesting fact we noticed was that the validation metrics outperformed training metrics, while uncommon, can occur due to several factors. These may include sampling variability in data splitting, where the validation set may have ended up with a slightly easier subset of samples to predict; regularization effects preventing overfitting; incomplete model convergence on training data, for instance, we see several convergence warnings for the LBFGS solver, which suggests that the model might not have fully converged on the training data, potentially leading to slightly suboptimal performance on the training set. Other reasons include differences in feature distribution between sets, presence of noise in training data, metric instability in imbalanced datasets, and others. However, the small differences observed (usually less than 1%) suggest good model generalization.

APPENDIX

Results for each variable on training and validation datasets:

Training Dataset				
		ROC AUC	F1 Score	Confusion Matrix
Variation 1	savings_acct	0.402916	0	[[706792, 0], [74, 0]]
	guarantees	0.261854	0	[[706850, 0], [16, 0]]
	current_acct	0.748522	0.786467	[[150022, 126938], [69026, 360880]]
	derivada_acct	0.777204	0	[[706603, 0], [263, 0]]
	payroll_acct	0.86457	0.000103	[[667859, 2], [39003, 2]]
	junior_acct	0.998986	0.141183	[[700283, 2], [6081, 500]]
	mas_particular_acct	0.825569	0	[[701121, 0], [5745, 0]]
	particular_acct	0.884666	0.214141	[[600002, 20131], [73919, 12814]]
	particular_plus_acct	0.813231	0	[[677357, 1], [29508, 0]]
	short_term_depo	0.924857	0	[[705857, 0], [1009, 0]]
	medium_term_depo	0.877228	0	[[705817, 0], [1049, 0]]
	long_term_depo	0.926317	0.3412	[[672212, 5132], [22394, 7128]]
	e_acct	0.860581	0.216472	[[640624, 7361], [50841, 8040]]
	funds	0.921595	0	[[694096, 3], [12767, 0]]
	mortgage	0.920838	0	[[702807, 0], [4059, 0]]
	pension	0.920788	0	[[700453, 1], [6412, 0]]
	loans	0.839712	0	[[705210, 0], [1656, 0]]
	taxes	0.859193	0.000784	[[668627, 17], [38207, 15]]
	credit_card	0.890356	0.000129	[[675775, 3], [31086, 2]]
	securities	0.91374	0.000227	[[689223, 12], [17629, 2]]
	home_acct	0.871426	0	[[704156, 0], [2710, 0]]
	pensions_2	0.861401	0.000094	[[664258, 2], [42604, 2]]
	direct_debt	0.868809	0.035647	[[615556, 1308], [88345, 1657]]

Variation 2	**savings_acct**	0.868903	0	[[706792, 0], [74, 0]]
	guarantees	0.964197	0	[[706850, 0], [16, 0]]
	current_acct	0.748034	0.786279	[[150207, 126753], [69288, 360618]]
	derivada_acct	0.880394	0	[[706603, 0], [263, 0]]
	payroll_acct	0.866318	0.000256	[[667856, 5], [39000, 5]]
	junior_acct	0.999425	0.881061	[[699481, 804], [766, 5815]]
	mas_particular_acct	0.841209	0	[[701121, 0], [5745, 0]]
	particular_acct	0.88482	0.229081	[[598646, 21487], [72734, 13999]]
	particular_plus_acct	0.813776	0	[[677356, 2], [29508, 0]]
	short_term_depo	0.945518	0	[[705857, 0], [1009, 0]]
	medium_term_depo	0.895236	0	[[705817, 0], [1049, 0]]
	long_term_depo	0.927018	0.350823	[[671929, 5415], [22090, 7432]]
	e_acct	0.86076	0.22182	[[640475, 7510], [50599, 8282]]
	funds	0.923179	0.004361	[[694052, 47], [12739, 28]]
	mortgage	0.926707	0	[[702806, 1], [4059, 0]]
	pension	0.922319	0.005588	[[700442, 12], [6394, 18]]
	loans	0.854446	0	[[705210, 0], [1656, 0]]
	taxes	0.859697	0.001202	[[668620, 24], [38199, 23]]
	credit_card	0.890593	0.00608	[[675713, 65], [30993, 95]]
	securities	0.914192	0.007435	[[689178, 57], [17565, 66]]
	home_acct	0.888943	0	[[704156, 0], [2710, 0]]
	pensions_2	0.862302	0.001266	[[664248, 12], [42579, 27]]
	direct_debt	0.869138	0.049997	[[614470, 2394], [87633, 2369]]

Variation 3	**savings_acct**	0.87294	0	[[706792, 0], [74, 0]]
	guarantees	0.973446	0	[[706850, 0], [16, 0]]
	current_acct	0.748993	0.786626	[[149726, 127234], [68714, 361192]]
	derivada_acct	0.880811	0	[[706603, 0], [263, 0]]
	payroll_acct	0.866259	0.001025	[[667849, 12], [38985, 20]]
	junior_acct	0.99957	0.885389	[[699470, 815], [706, 5875]]
	mas_particular_acct	0.841194	0	[[701121, 0], [5745, 0]]
	particular_acct	0.884887	0.230861	[[598531, 21602], [72596, 14137]]
	particular_plus_acct	0.813682	0	[[677358, 0], [29508, 0]]
	short_term_depo	0.94728	0	[[705857, 0], [1009, 0]]
	medium_term_depo	0.895238	0	[[705817, 0], [1049, 0]]
	long_term_depo	0.927018	0.350078	[[671952, 5392], [22114, 7408]]
	e_acct	0.861151	0.220272	[[640548, 7437], [50673, 8208]]
	funds	0.92312	0.004206	[[694055, 44], [12740, 27]]
	mortgage	0.927031	0	[[702807, 0], [4059, 0]]
	pension	0.922354	0.005587	[[700441, 13], [6394, 18]]
	loans	0.854412	0	[[705210, 0], [1656, 0]]
	taxes	0.859777	0.001202	[[668623, 21], [38199, 23]]
	credit_card	0.890683	0.006335	[[675711, 67], [30989, 99]]
	securities	0.914217	0.007435	[[689178, 57], [17565, 66]]
	home_acct	0.889242	0	[[704156, 0], [2710, 0]]
	pensions_2	0.862506	0.001032	[[664248, 12], [42584, 22]]
	direct_debt	0.869315	0.046836	[[614537, 2327], [87788, 2214]]

Validation Dataset				
		ROC AUC	F1 Score	Confusion Matrix
Variation 1	savings_acct	0.448868	0	[[200325, 0], [8, 0]]
	guarantees	0.117191	0	[[200328, 3], [2, 0]]
	current_acct	0.756396	0.789899	[[39830, 38322], [17412, 104769]]
	derivada_acct	0.750282	0	[[200289, 0], [44, 0]]
	payroll_acct	0.859517	0.007521	[[192120, 1999], [6183, 31]]
	junior_acct	0.99722	0.127197	[[199214, 0], [1043, 76]]
	mas_particular_acct	0.886404	0	[[197523, 3], [2807, 0]]
	particular_acct	0.926977	0.245744	[[180445, 4825], [12277, 2786]]
	particular_plus_acct	0.858969	0.028845	[[194046, 1811], [4384, 92]]
	short_term_depo	0.912585	0	[[200006, 3], [324, 0]]
	medium_term_depo	0.897696	0	[[200152, 0], [181, 0]]
	long_term_depo	0.935919	0.293059	[[193029, 2643], [3407, 1254]]
	e_acct	0.876874	0.168256	[[187944, 2829], [8422, 1138]]
	funds	0.935718	0.058437	[[197675, 571], [2007, 80]]
	mortgage	0.94664	0	[[199771, 41], [521, 0]]
	pension	0.932103	0.037847	[[198674, 780], [847, 32]]
	loans	0.858123	0	[[199995, 0], [338, 0]]
	taxes	0.865688	0.03068	[[193529, 1867], [4831, 106]]
	credit_card	0.918365	0.040483	[[194089, 1880], [4235, 129]]
	securities	0.928463	0.085966	[[195880, 1401], [2852, 200]]
	home_acct	0.899709	0	[[200015, 0], [318, 0]]
	pensions_2	0.858688	0.009136	[[191616, 1993], [6684, 40]]
	direct_debt	0.870896	0.051411	[[182140, 2092], [15621, 480]]

Variation 2	**savings_acct**	0.933433	0	[[200325, 0], [8, 0]]
	guarantees	0.988746	0	[[200331, 0], [2, 0]]
	current_acct	0.756892	0.789503	[[39858, 38294], [17517, 104664]]
	derivada_acct	0.90797	0	[[200289, 0], [44, 0]]
	payroll_acct	0.860161	0.008951	[[192103, 2016], [6177, 37]]
	junior_acct	0.998574	0.827032	[[199092, 122], [244, 875]]
	mas_particular_acct	0.885058	0	[[197474, 52], [2807, 0]]
	particular_acct	0.925391	0.253438	[[179507, 5763], [12041, 3022]]
	particular_plus_acct	0.861638	0.029767	[[193979, 1878], [4380, 96]]
	short_term_depo	0.945722	0.005831	[[199991, 18], [323, 1]]
	medium_term_depo	0.927063	0.037736	[[199969, 183], [174, 7]]
	long_term_depo	0.938429	0.303536	[[192704, 2968], [3296, 1365]]
	e_acct	0.873791	0.171671	[[187851, 2922], [8388, 1172]]
	funds	0.938061	0.06108	[[196365, 1881], [1962, 125]]
	mortgage	0.948643	0.00595	[[197987, 1825], [514, 7]]
	pension	0.93383	0.031701	[[197539, 1915], [834, 45]]
	loans	0.887794	0	[[199576, 419], [338, 0]]
	taxes	0.865983	0.03113	[[193439, 1957], [4828, 109]]
	credit_card	0.920199	0.044042	[[194071, 1898], [4223, 141]]
	securities	0.930085	0.096486	[[195539, 1742], [2809, 243]]
	home_acct	0.913726	0.009112	[[199896, 119], [316, 2]]
	pensions_2	0.85963	0.010488	[[191607, 2002], [6678, 46]]
	direct_debt	0.870889	0.060638	[[181975, 2257], [15527, 574]]

Variation 3	**savings_acct**	0.931511	0	[[200325, 0], [8, 0]]
	guarantees	0.91059	0	[[200315, 16], [2, 0]]
	current_acct	0.758703	0.790143	[[39445, 38707], [17107, 105074]]
	derivada_acct	0.897898	0	[[200284, 5], [44, 0]]
	payroll_acct	0.864173	0.008957	[[192108, 2011], [6177, 37]]
	junior_acct	0.999552	0.840149	[[199085, 129], [215, 904]]
	mas_particular_acct	0.890196	0	[[197523, 3], [2807, 0]]
	particular_acct	0.930324	0.257369	[[181266, 4004], [12247, 2816]]
	particular_plus_acct	0.866979	0	[[195857, 0], [4476, 0]]
	short_term_depo	0.943757	0.012579	[[199701, 308], [320, 4]]
	medium_term_depo	0.928459	0.035794	[[199894, 258], [173, 8]]
	long_term_depo	0.940121	0.30518	[[192758, 2914], [3297, 1364]]
	e_acct	0.879096	0.170943	[[187857, 2916], [8394, 1166]]
	funds	0.942005	0.059712	[[197506, 740], [2000, 87]]
	mortgage	0.951792	0.00789	[[198818, 994], [515, 6]]
	pension	0.939714	0.036748	[[198570, 884], [846, 33]]
	loans	0.887644	0	[[199658, 337], [338, 0]]
	taxes	0.873381	0.023401	[[194167, 1229], [4864, 73]]
	credit_card	0.923466	0.04375	[[194073, 1896], [4224, 140]]
	securities	0.931958	0.097025	[[195567, 1714], [2809, 243]]
	home_acct	0.917353	0	[[200015, 0], [318, 0]]
	pensions_2	0.861479	0.010707	[[191601, 2008], [6677, 47]]
	direct_debt	0.870459	0.058622	[[181953, 2279], [15546, 555]]