Industry Project Final Report (UBS O'Connor)

by

Xiaoya Wang
Zheyu Gu

1. Executive Summary

The main purpose of this project is to discover the effectiveness of using AutoML technology in modeling different kinds of datasets. At the beginning of this report, we will first introduce what AutoML is and the two major AutoML systems we have used for this project. Diving into the world of data, we started getting familiar with the two major AutoML vendors, H2O and SparkBeyond, by trying out the simple Iris dataset, and it turned out that the simple dataset was not able to differentiate AutoMLs from human-built machine learning models. Coming to the second phase of the project, we implemented AutoML models on more complex panel datasets and discovered other techniques that may be useful for this kind of dataset, which include panel regression and hierarchical modeling. Thanks to their powerful feature engineering functionality, the AutoMLs eventually won the crown when the datasets were becoming harder to predict. In the end, we attempted to model financial panel data by using AutoMLs, panel regression, and hierarchical modeling within limited time, but the low signal-to-noise ratio of financial data made the process more challenging and we had to sacrifice the accuracy for shorter training time. The advantages and drawbacks of AutoMLs and other major techniques discovered throughout the project will also be discussed in this report.

By doing this project, we personally experienced the strength of AutoML. Even though there are still areas for AutoML technology to improve on, we believe AutoML already possesses the capability of making data science easier and more accessible to a larger group of people, and we hope to see it to become the mainstream of future world, just like what machine learning has been doing recently.

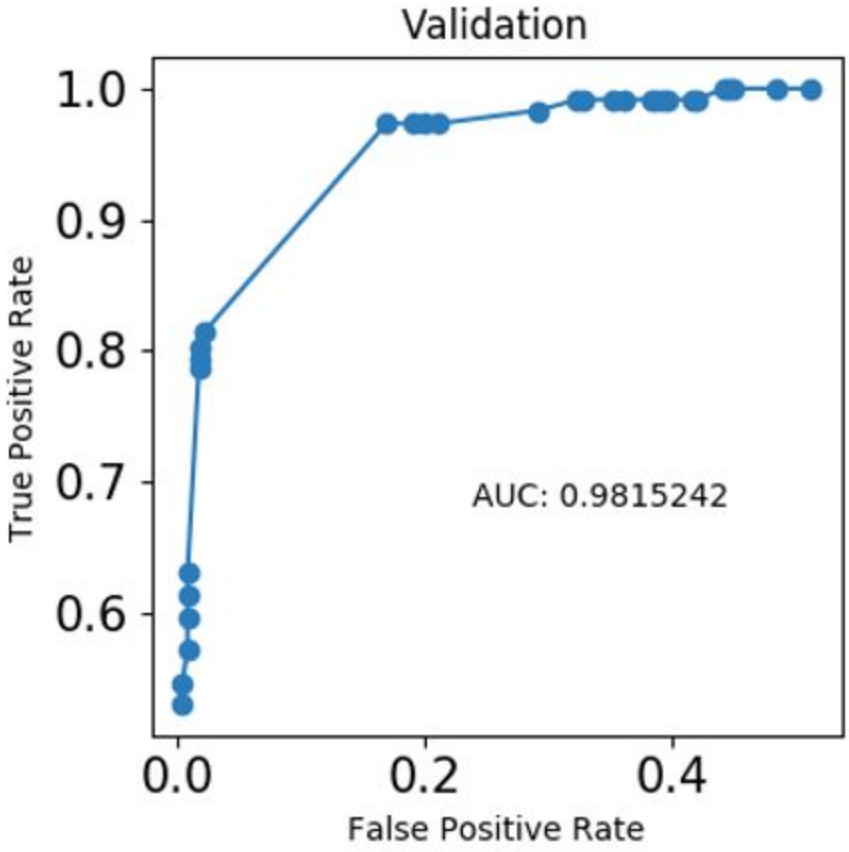
2. Introduction to AutoML and Its Relationship with Machine Learning

"Machine learning" has recently been a hot term used by people in different fields, and generally it represents a process where computer algorithms try to make predictions based on some known information. For example, machine learning algorithms can filter spam emails by detecting some kinds of features of emails, or in the financial world, machine learning algorithms can also help banks to predict whether a person will default on his/her loans based on personal information. Due to the power and versatility of machine learning, not only people who are in technology giants but also other people in different fields hope to implement it into their businesses. However, not everyone is proficient in coding and constructing machine learning pipelines, and so it is one of the reasons why Automated Machine Learning (AutoML) comes to the world. With well-established AutoML platforms, users only need to click a few buttons on their user interfaces to let the system know what the users are trying to achieve without writing a single line of code, and then the result of machine learning models will be available to them. Indeed, to better control the AutoML systems, users need to have some basic understandings of data science knowledge, for example what cross-validation means, what each performance metric represents, what classification or regression is, etc..

Usually a machine learning pipeline contains the following sections: data collection and preprocessing, feature engineering, model training and validation, and prediction and visualization. In data preprocessing, AutoML is able to handle most of the data issues that humans can expect to have, saving people's time for more intellectually-required tasks. For example, AutoMLs will try log-transformation when they automatically detect skewd targets,

or AutoMLs will transform detected categorical features to dummy variables for later use. In most of the cases, original data features are not sufficient for machine learning models to come up with accurate predictions, so usually feature extraction and engineering are needed before training models. When AutoML is not available, humans often need to spend some time thinking about the underlying problems and trying to figure out what some useful features might be to explain the variance of target variables. On the other hand, without the need to understand contexts, AutoMLs perform feature importance check and feature engineering while tuning models in iterative manners, and features in future iterations will be updated based on previous iteration results. Hence, feature engineering from humans is more like an intuitive process, while feature engineering from AutoMLs is more like an empirical process. In tuning models, AutoMLs try several different machine learning algorithms based on problem types, and hyperparameter tuning is done by quickly running model validations. Final AutoML models will be chosen from best-performing feature engineering, ML models, and hyperparameter combinations. In the final step, AutoMLs also generate predictions both in-sample and out-sample with corresponding evaluation metrics. Some plots which may help users to interpret models will also be provided by systems, and an example of the visualization can be referred to *Figure 1*.

Figure 1: ROC Curve Generated by H2O AutoML System on Iris Dataset



Through our observation, AutoMLs' main advantages are their usability, automated feature engineering, and automated model and hyperparameter tunings. Despite the convenience and usually better performance provided by AutoMLs, they do have some drawbacks that users need to be aware of. AutoMLs are the paradises for non experts in data science fields, but they could also be troubles for technical people, who might be thwarted to achieve more advanced goals due to the fact that little knowledge on what is happening behind systems is revealed. In addition to that, some AutoML models running on complex datasets may require significantly long running time as models are trying to find optimal features, algorithms, and hyperparameters through iterations over complexity. Last but not

the least, AutoML systems are only accessible to people who bought licenses from the vendors, so considerations such as cost and marginal benefit of AutoML need to be taken care of in business settings.

3. H2O and SparkBeyond AutoML Platforms

H2O.ai and SparkBeyond are two leading AutoML companies, keeping adding cutting-edge technologies and algorithms to the world AI field. With the belief that AI makes work easier and more efficient, they keep supporting various organizations with their platforms to help them solve problems or make innovations. Both of them are offering Python packages, so users are able to realize all functions by code only. This functionality enables people to preprocess data, analyze data, build machine learning models, get predicted results, and deal with the results in one package assignment. Members from H2O and SparkBeyond both have attended our meetings and offered us some useful ideas about how to make the AutoML work as we wish.

H2O platform

H2O driverless AI platform contains many machine learning workflows, including feature engineering, model validation, model tuning, model selection, and model deployment. With detailed tutorials, users could adjust the H2O platform to make it achieve the goal easily and satisfyingly. There are plenty of algorithms available in H2O, and for every dataset H2O would train them and select out the most accurate one; the common supervised algorithms include Neural Networks, GLM, GBM, Naive Bayes, SVM, and XGBoost, while the common unsupervised learning models include K-Means, Aggregator,

and PCA. Data preparation is always the most important part in training machine learning models: garbage data may lead to a garbage model. H2O could do many transformations on the raw data to make it more appropriate to analyze, and this process is called feature engineering. Every time H2O builds machine learning models, they first do feature engineering, create and add new features to models, and identify the ones that are useful to models. Transformation techniques used by H2O include scaling, decomposition, and aggregation. Then H2O will choose and train suitable algorithms based on the size, structure, and type of the user's dataset, and based on their performances, H2O could find out the best fitted model. Every time before the training starts, users could set different training levels, including accuracy, time, and interpretability. Besides that, users could customize their own machine learning processes by adding more setups, such as classification or regression, the scorer he/she wants H2O to compare, and the number of CV folds for training and testing. Based on these settings, H2O would produce an integrated report of training results, including the model tuning, top features, and the final model details.

SparkBeyond

Compared to the H2O platform, SparkBeyond takes much shorter time to train models. Before the learning process, SparkBeyond enables users to first run the feature discovery and have a first look at the dataset; during this process, users could even link some context datasets to their main dataset to see the relevance between these two datasets. When users are not satisfied with the training results, they could try Rebuild Model capability which allows users to better explore and analyze the model: users can choose features they want to use, and tune the parameters of the model by themselves. However, unlike those used by H2O,

features used by SparkBeyond are all binary, which sometimes are too many and not effective enough.

4. AutoML vs. Machine Learning on Classical Datasets

Trying to understand how AutoML performs against traditional machine learning methods, we started off with the most classical machine learning data, Iris dataset, which is aimed to classify between three species of flowers based on their features of sepals and petals. In this analysis, we first attempted to construct high-performing human-built machine learning models, and we then compared the results with the ones generated by H2O and SparkBeyond. Even though the AutoML models did not outperform the machine learning models built by us, we were not really surprised by the result due to the fact that Iris dataset is a relatively easy task.

During the preliminary analysis of this dataset, we deployed the LGBM model to find out feature importance. As shown in *Figure 2*, petal length and petal width generally have greater impacts on classification results, which somewhat coincides with the feature importance graph from the LGBM model shown in *Figure 3*. Knowing what features are more important in modeling, we then proceeded to the model construction stage.

Figure 2: SHAP Value of Different Features

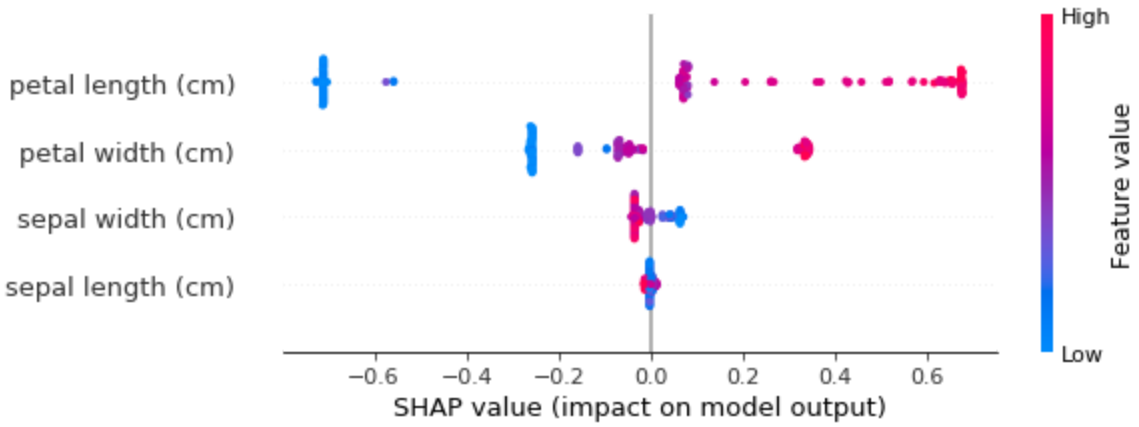
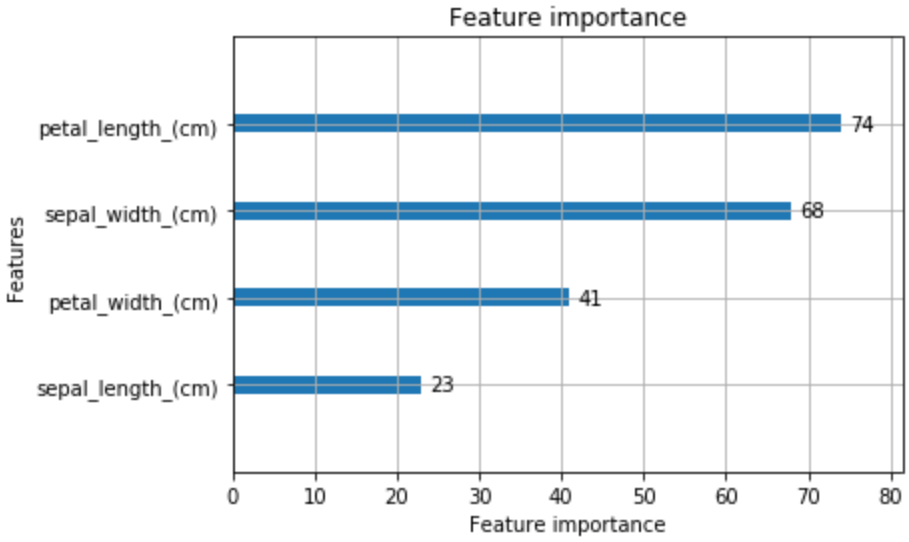


Figure 3: Feature Importance from LGBM



Our methodology of building machine learning models is to tune a few models suitable for classification tasks, and we take the most votes across different machine learning models to be the final prediction of our ensemble model. After doing thorough selection process of machine learning models, we decided to rely on the following 7 models: kNN, Naive Bayes, Logistic Regression, Random Forest, Gradient Boosting Tree, Support Vector Machine, and Artificial Neural Networks, which are not only easy to implement but also have good prediction ability.

Optimal hyperparameters were discovered by grid searches over 10-fold cross validation. Listed in *Table 1* is the summary of individual models' hyperparameters and cross-validation F1 score. Individual machine learning models are showing high accuracy in this classification task, with 10-fold cross-validation F1 scores all around 0.95.

Table 1: Summary of Individual ML Models' Hyperparameters and Performance

Model	**Hyperparameter(s)**	**F1 Score (10-fold CV)**
kNN	n_neighbors=7	0.9523
Naive Bayes	N/A	0.9614
Logistic Regression	N/A	0.9608
Random Forest	n_estimators=100; max_depth=4	0.9530
Gradient Bossting Tree	n_estimators=100; learning rate=0.1	0.9356
Support Vector Machine	N/A	0.9689
Artifical Neural Networks	solver='lbfgs'; alpha=1e-5 hidden_layer_sizes=(5,5)	0.9593

Again, H2O and SparkBeyond are the two major AutoML systems we would like to discover in this project. In an H2O project, the most important settings that require users to input are "Accuracy" (controls accuracy needs of the model), "Time" (controls duration of the experiment), and "Interpretability" (controls complexity of the model). For this specific project, due to the simplicity of the data and limited time allowed for H2O trial edition, we have set accuracy to be 7, time to be 1, and interpretability to be 8, with maximum values of all items being 10. As noted in H2O report, some feature engineering and feature transformations have

been done during the training process, and top 3 features used by H2O models are: original petal length, mean of the target column grouped by petal length, petal length minus sepal width, which again matched with what was discovered by LGBM model from our previous analysis. Unlike H2O, there is not too much for users to worry about in order to initiate a SparkBeyond experiment besides uploading a dataset and identifying what they would like to predict. Without too much surprise, top 3 features (only binary features) generated by the SparkBeyond system are petal length < 2.45, ceil(petal length) == 6.0, and floor(petal width) == 1.0. In terms of the training time, SparkBeyond's advantages are salient. H2O took almost 19 minutes to complete this classification task, while SparkBeyond only took 16 seconds to complete. Coming to the battle between human-built models and AutoML models, we noticed there is no significant difference in F1 score, and the score report is shown in *Table 2*.

Table 2: Comparison Between Human-Built Models and AutoML Models

Model	**F1 Score (10-fold CV)**	**Ranking (1-4)**
Best of Individual ML Models	0.9689	1
Human-Built Ensemble Model	0.9496	3
H2O AutoML Model	0.9546	2
SparkBeyond AutoML Model	0.9390	4

Even simplest models such as kNN and Naive Bayes can generate predictions with high accuracy, we believe the slight difference between human-built models and AutoML models might just come from randomness. Therefore, it is still too early to conclude whether AutoML begins to revolutionize the data science world only with this easy-to-predict dataset, but one

insight that may be useful for readers is that AutoML models are not really necessary on simple datasets like this. In the next section, we will put our attention on panel datasets, and we are going to explore the effectiveness of AutoML in modeling complex panel datasets by comparing them with other popular panel modeling techniques.

5. AutoML vs. Other Modeling Techniques on Retail Panel Datasets

AutoML's effectiveness cannot be tested on simple datasets like Iris, so we decided to move on to more complicated datasets. To be more relevant to financial modeling, we implemented AutoML models on panel datasets and then compared the results with other panel data modeling techniques, which include panel regression and hierarchical modeling. This time when the prediction process has more noises, AutoML performance finally does not let us down.

Time-series data, as its name suggests, includes data points indexed in time order, while cross-sectional data involves data points across different entities. In practice, sometimes modeling has to be done both from time-series and cross-sectional points of view, and we call this data type which combines features of the time-series and cross-sectional data to be panel data. The two datasets we have used in this phase of the project are Walmart sales and Rossman sales dataset. In Walmart dataset, sales time-series data is detailed for each department of each store. As an example, we have Department # 2 of Walmart Store #3 sales data running from February 2010 to October 2012. Except for sales data itself, we also have other time-related and time independent features including isHoliday (whether the corresponding week was a holiday), temperature, fuel price, markdowns, CPI, unemployment, store type, and store size. With all of the information available as of today, the goal of this dataset is to predict the next day's sales for

each department of each store. With the same goal as Walmart dataset, Rossman has slight differences in its data structure and features. Unlike Walmart dataset having cross-sectional entities in both store and department, Rossman only has store-wise data, and its features include day of the week, open (indicating whether the store is open), promotion, customers, StateHoliday (whether the day is a state holiday), SchoolHoliday (whether the day is a school holiday), store type, assortment level, and competition distance. Even though panel data is widely used in finance, such as stock price time-series data across different stocks, we still would like to start with something with a higher signal-to-noise ratio for testing purposes. Once we have a better understanding of AutoMLs' capacities, we may implement similar techniques on financial panel data but with the need for more tuning on models.

In order to build a comparison baseline, we started with the simplest predictions by using naive mean and fast-built machine learning models, which include linear regression and LGBM. Even though simple machine learning models are easier for us to understand and implement, we also have to do some additional work on manually separating entity groups and manually creating time-series related features (such as lag sales and EWMA), which might not be sufficient enough to come up with powerful models. Running an LGBM model gives us the feature importance of Walmart and Rossman dataset as shown in *Figure 4* and *Figure 5* respectively.

Figure 4: Feature Importance of Walmart Sales from LGBM

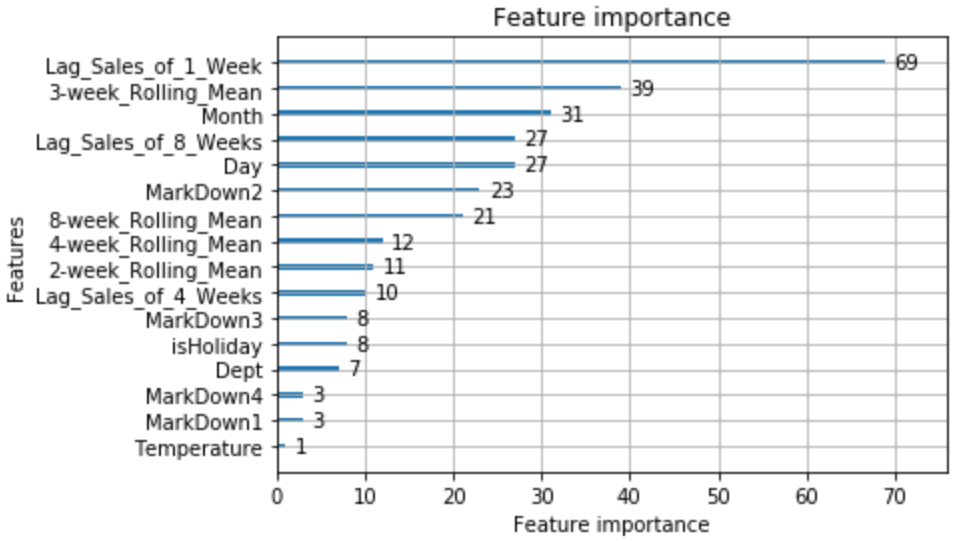
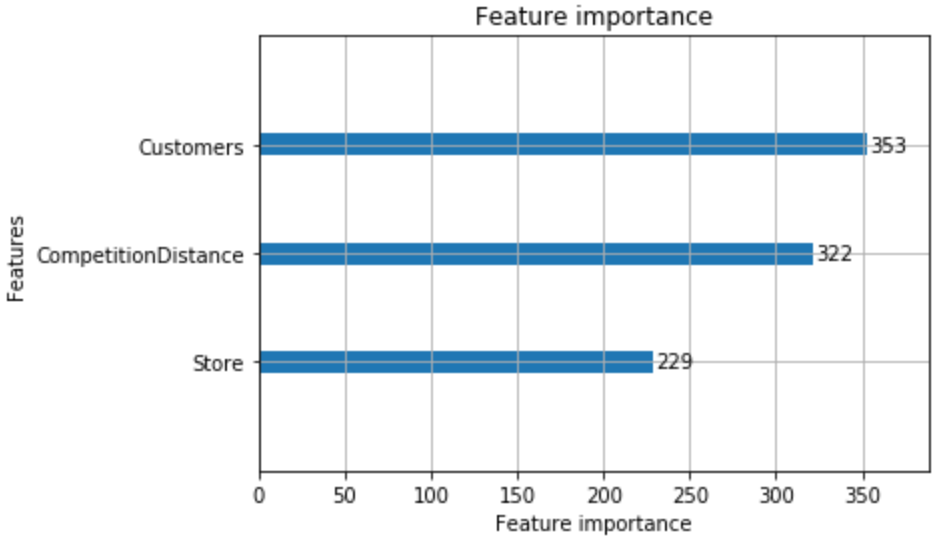


Figure 5: Feature Importance of Rossman Sales from LGBM



Panel regression is nothing more than the OLS with slight modifications under cross-sectional groups automatically considered, and we mainly used three panel regression techniques which are PooledOLS, PanelOLS, and RandomEffects. PooledOLS has the format $y_{it} = \beta x_{it} + \varepsilon_{it}$ where i stands for i_{th} entity group and t stands for t_{th} time step, totally ignoring

the entity effect. Two-way PanelOLS has the format $y_{it} = \alpha_i + \gamma_t + \beta x_{it} + \varepsilon_{it}$, taking both entity effect and time effect under consideration. RandomEffects has the format $y_{it} = \beta x_{it} + \mu_i + \varepsilon_{it}$, considering the entity effect to be random and using quasi-difference to efficiently estimate parameters. With additional functionalities added on OLS, panel regression is still far away from being the optimal model in panel data settings. Although users do not have to manually separate entity groups before running panel regression, inefficient feature engineering is still an outstanding problem because panel regression will not generate any additional features to users. Panel regression adds the entity effect and the time effect as "constant" components into the OLS model, but it still ignores the potential fact that relationships between features and targets in different groups may be different (i.e. panel regression uses same variable slopes across groups). With these drawbacks, panel regression did not show us with a more brilliant performance over the previous linear regression model which may include regularization terms to deal with collinearity.

Hierarchical modeling is another traditional method to model panel data, and one can consider it as the midway between pooled measurements (where one big regression is run regardless of groups) and unpooled measurements (where individual regression is run on each group). In other words, hierarchical modeling assumes different groups share some similarity but their features still have different relationships with the target, so same parameters in different groups are drawn from one distribution. The distribution-based hierarchical modeling provides both target and feature parameters' distribution after the model has been sampled, and *Figure 6* and *Figure 7* respectively show the comparison between observed sales distribution and posterior predicted sales distribution of Walmart and Rossman dataset. The main idea of hierarchical

modeling is plausible, but there are indeed a great amount of obstacles that users need to overcome for accurate models. As illustrated in *Figure 8*, one of the major obstacles is that people usually do not have relatively accurate prior knowledge about the dataset for setting appropriate distributions and hyperpriors, and the result is just a poor model due to inaccurate feature parameters' distribution. Possibly due to this reason, we observed the instability of the hierarchical model. In our first experiment of Walmart dataset, the test set performance was surprisingly better than in-sample and AutoML performance, so we decided to increase our test set from last 10% of the entire data to last 30% of the entire data ordered by time. The test set performance eventually turned out to be even worse than the naive mean prediction in our second attempt. With more detailed look into the result, we found that most of the out-sample predictions were showing really good results (prediction error below 5000) and the proportion of samples having prediction error greater than the RMSE of 63602.61 was only 0.2%. Hence, hierarchical modeling will lead to unstable predictions (really good predictions on certain groups while really poor predictions on other groups) when inaccurate distribution and hyperprior assumptions are made. In addition to this problem, hierarchical modeling also has extremely long running time. Walmart dataset took almost 10 hours (100 samples in each chain) to complete and Rossman dataset took about 5 hours (20 samples in each chain) to complete, and the number of chains we selected have already been much lower than what was recommended. These two major obstacles along with others have made hierarchical modeling not to be the priority when people want to model complex panel dataset in which some groups behave largely differently than other groups.

Figure 6: Observed and Predicted Sales Distribution of Walmart Sales

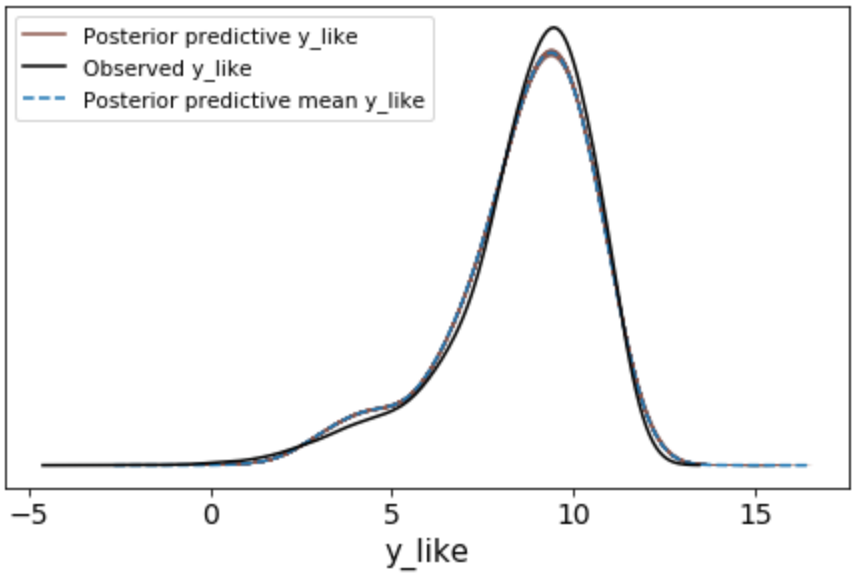


Figure 7: Observed and Predicted Sales Distribution of Rossman Sales

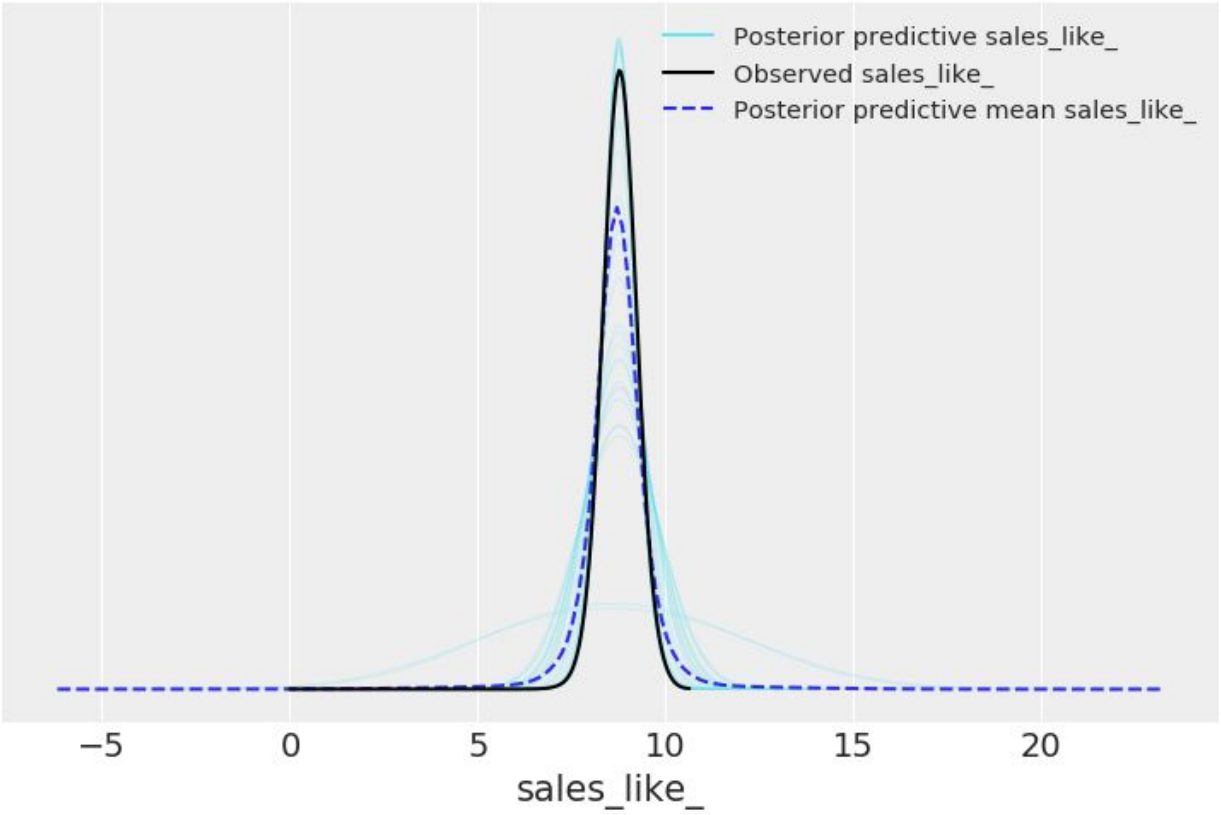
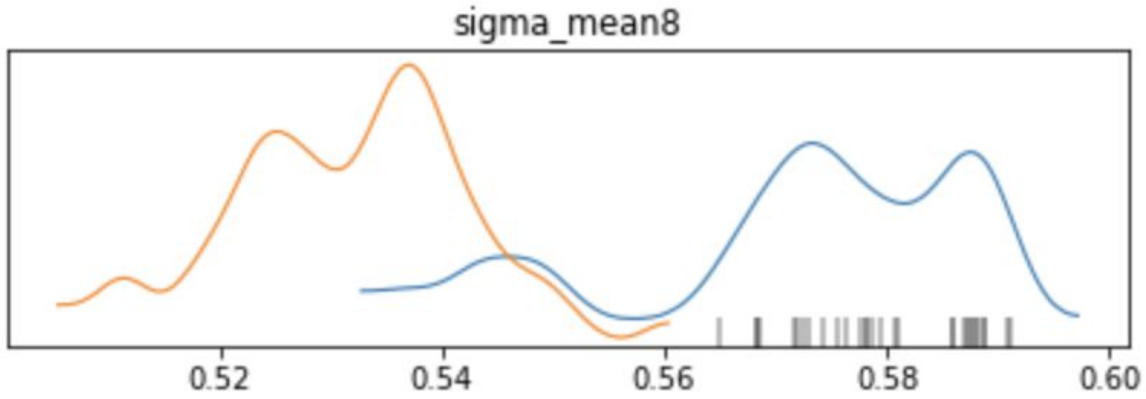


Figure 8: Observed and Predicted Sigma of 8-Week Rolling Mean of Walmart Sales



Both H2O and SparkBeyond have powerful functionality of handling panel data. H2O's "Time Group" feature and SparkBeyond's "Key Column" feature are used to identify the cross-sectional aspect of panel data. Both platforms offer time-series specific feature engineering (e.g. lag features, moving average features, day of week, holiday, max, min, etc.), and again the only difference is that SparkBeyond only generates binary features. In regards of unique offering from the two platforms, H2O has a functionality of automatically detecting gap and forecast horizon based on the position and length of the test set, and SparkBeyond can incorporate some more advanced features like seasonality into modeling. Because AutoMLs' results are just a few button-clicks away, we are just going to directly compare AutoMLs' top features and model performances with methods mentioned in the following tables.

Table 3: Feature Comparison of Walmart Sales

LGBM	**SparkBeyond**	**H2O**
1-week lag sales	Percentile of last 10 weeks average sales < 0.45	EWMA of sales lags (28, 32, 36, 40, 44, 48, 52, 56, 60, 64)
MA of sales for 3 weeks	Percentile of exponential of last week's sales < 1.42e-8	EWMA of sales lags (32, 40, 44, 45, 51, 52, 53, 60, 104, 105)

Month	8-week lag sales > 19713.77	52-week lag sales

Table 4: Feature Comparison of Rossman Sales

LGBM	**SparkBeyond**	**H2O**
Customers	Customers < 41	EWMA of sales lags (28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105, 112) grouped by store
Competition distance	HistoricalAverage(sales of store between all past days and 3 days before "today") ≥ 5344.945	EWMA of sales lags (28, 35, 42, 49, 56, 63, 77, 91, 105, 119) grouped by store
Store	Promo ≠ 0	Open

Table 5: Performance Comparison of Walmart Sales

Model	**In-Sample RMSE**	**CV RMSE**
Naive Mean	24117	N/A
Linear Regression	6840.83	6924.16
LightGBM	6413.27	6522.35
H2O AutoML	N/A	5992.58
SparkBeyond AutoML	N/A	5007.07
PooledOLS	6866.51	N/A
PanelOLS	6874.59	N/A
RandomEffects	6866.51	N/A
First Hierarchical Attempt	13781.73	4012.49
Second Hierarchical Attempt	11027.32	63602.61

Table 6: Performance Comparison of Rossman Sales

Model	In-Sample RMSE	CV RMSE
Naive Mean	3852.53	N/A
Linear Regression	1453.72	1575.06
LightGBM	1099.36	1655.05
H2O AutoML	N/A	924.78
SparkBeyond AutoML	N/A	558.87
PooledOLS	1453.96	1513.47
PanelOLS	1453.96	1513.47
RandomEffects	1486.78	1508.24
Hierarchical	2106.04	2721.16

As obviously demonstrated in *Table 5* and *Table 6*, two AutoMLs finally outperforms other models this time on more complex datasets, and specifically SparkBeyond beats H2O in both of the datasets. We think the major difference happens in the feature engineering stage, as the models used by AutoML platforms are similar to what we used for machine learning, like LightGBM used by H2O for Walmart sales. Shown in *Table 3* and *Table 4*, the features generated by two AutoMLs are too specific for humans to discover, and the superior performances of AutoMLs have proven the validity of those "impossible" features. In the next section, we are going to reveal the results we got when we quickly implemented all of the above models on a financial panel dataset less than a week.

6. AutoML vs. Other Modeling Techniques on a Financial Panel Dataset

After dealing with the traditional retail panel data using multiple techniques, we try to move on to another type of panel time-series data - financial panel time-series data. They are very similar in their structures, so utilizing the same methods on them should be very efficient and can lead to satisfying results.

The dataset we choose is from Kaggle website, called "New York Stock Exchange", containing S&P 500 companies historical prices. Learning historical data and predicting the price for the next day/month/year is always the main and most difficult task in the financial field. Multiple random effects exist in the financial world, so there does not seem to be a stable and effective predicting model. Quantitative researchers, financial data scientists and stock traders are all very interested in finding a rule from the historical data, so that they could predict future prices/returns based on it and make money. Besides the price features, such as Open, Close, Low, High, Volume, the dataset also contains fundamental information and securities of each company. More detailed information often leads to more accurate predictive models, while complicated models may sometimes turn out to be ineffective or overfitting. Therefore, in some of the models below, we did not use all company information, while we use prices information in all of them.

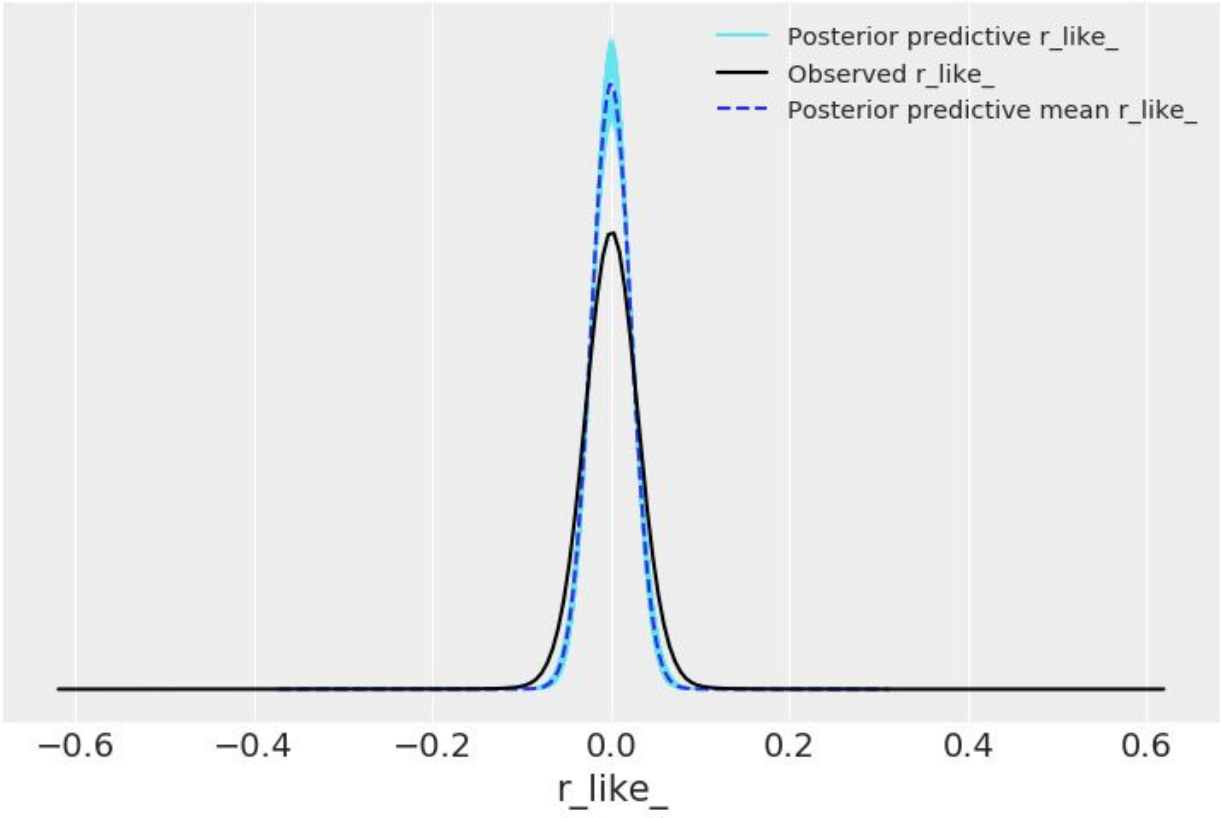
In this project, we choose to forecast the next day's return of each stock, instead of the close price; the reason for our option is that returns can be comparable to each other, while prices depend on the previous day's prices.

Just like what we did for sales panel data, we manually added some time-series features (Lag returns and the EMWA) for our human-built models (Panel regression models and basic models). Due to the features of PyMC3, we applied normalization scaling to all of our features before

building the hierarchical model; although we dropped several less important and dummy features to simplify our data, it still took our nearly 24 hours to get the final model. Running it through H2O is also a little bit slow, so we set the time of building it to 1, and this may make the results not good enough.

From the feature engineering result of H2O, we could easily get the important pattern of our dataset, and know the importance of each feature. The top 1 feature is today's return, so we can conclude that next day's return is much related to today's value. H2O has tried LightGBM, XGBoost GBTree, and XGBoost GBLinear, and the final model chosen by H2O is the LightGBMModel.

Figure 9: Observed and Predicted Distributions of the Next Day's Return



The observed and predicted stock return distributions of S&P 500 are shown above in *Figure 9.* For a well-fitted model, the patterns of observed and predicted data on the plot should be very similar, while our result is not good enough. The reason for this may be because we only generate 400 samples and 400 interactions to tune; we could train a better model if we have a high-performance computer system.

Table 7: Performance Comparison of Rossman Sales

Model	**In-Sample RMSE**	**CV RMSE**
Naive Mean	0.0179	N/A
Linear Regression	0.0178	0.0203
LightGBM	0.0179	0.0200
H2O AutoML	0.0153	0.0195
PooledOLS	0.0179	0.0200
PanelOLS	0.0179	0.0200
RandomEffects	0.0179	0.0200
Hierarchical	0.0185	0.0201

As shown in *Table 7*, the results of all techniques on this dataset are very close to each other; although the H2O AutoML has a best performance, its difference from others' performances is not significantly big due to purposefully chosen short training time. Because the time taken by the Hierarchical is so long, using it for such a complicated dataset may not be a wise choice.

7. Reflection and Conclusion

AutoML field is a totally new field to us, so we find exploring it and utilizing it to achieve our goals are not only interesting but also very helpful to our future career. With its easily handled and comprehensive functions, even non-experts are able to get their desired results. Also, some complicated datasets which are difficult to analyze can be a simple task to AutoML platforms. Although we have many other packages that enable us to build machine learning models, preprocessing data and choosing the best of them are always the most difficult part. We believe that in the future, as the AutoML platforms become more and more practical and advanced, almost all companies would find it useful, just like they find data science useful now.

As to the classical and simple dataset, some simple human-built models would be efficient enough, but AutoML may choose the other complicated but not suitable models. However, when it comes to some complicated datasets, such as panel time-series dataset, NLP dataset, recommended dataset, and many others, AutoML would use less time, be easily implemented, and turn out to good results.