Final Report: Novartis Project

Forecast Sales of Pharmaceutical Products

Hang Xu (hx2321), Senqi Zhang (sz3016), Shuyue Xu (sx2316), Yajie Zhang (yz3876), Zehui Wu (zw2804)

Abstract

The post-approval lifecycle of a drug encompasses a series of distinctive events in drug sales, and it is hard to model these lifecycle transitions using traditional time series approaches because they only consider past historical trends thus leading to an over/underestimation of revenue. This project aims to increase prediction accuracy by identifying potential transitions between lifecycle stages. In collaboration with Eric from Novartis, we first applied multiple machine learning models for predicting transition dates and then used regression models to tune the benchmark we created based on clusters' historical trends. In addition, we also applied a Recurrent Neural Network model to forecast the sales without considering the prediction of transition points. In conclusion, the RNN achieved a 7.14% MAE lift compared to the benchmark while the original method only achieved a 0.5% lift.

1. Introduction

Sales volume forecasting for pharmaceutical companies is often tricky, as drugs in different stages often have varying sales performance. Connelly & Daignault (1) proposed a simple long-term sales forecasting model based on the product life cycle concept, but we still want to build a model that is specific to drugs.

In this project, we worked with Eric from Novartis to improve the benchmark by recognizing possible transitions between lifecycle stages. This project was divided into two stages, the first stage was to predict whether one drug will transition from an increasing growth stage to a stable maturity stage based on historical data and features we created to reflect current market conditions. If a transition point occurred in the prediction period, we need to accurately forecast the time it occurred. The second stage is to predict the sales volume based on the features we generated and the predicted transition point we got from the first stage. The following figure demonstrates the whole process of predicting the transition date and sales.

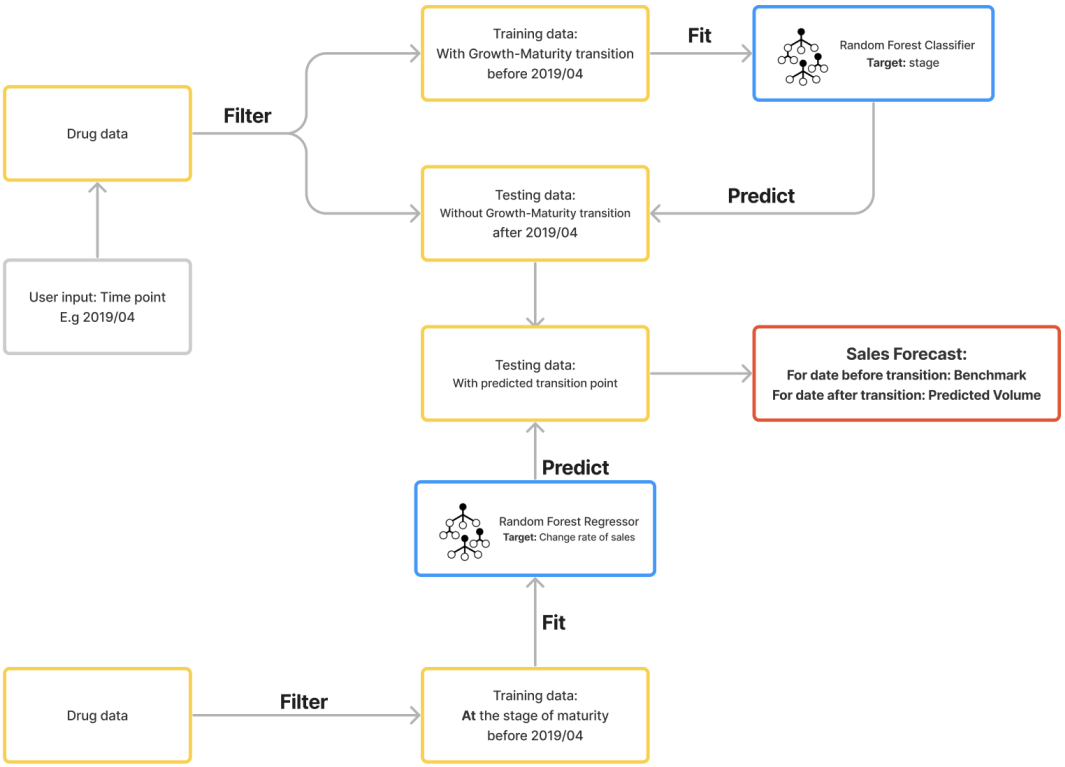


Figure 1: model process

2. Data Description

This project used anonymized data provided by Novartis, which consists of 1865 products. Novartis provided one main table and several auxiliary tables. The main table contains features on drug characteristics. The following is the data description for each variable of the table.

`Country` - Anonymized name of the country. Each country has a different portfolio of cluster/drugs.

`Cluster` – Anonymized name of the drug. Different countries sell the same drug, and usually they are at the same Lifecycle stage (not always since different compliance rules in each country may lead to different launch dates of the drug)

`Cluster_id` – Unique identifier corresponding to a particular country-cluster combination.

`Date` – Date of the month, always has the format DD/MM/YYYY. Since we work with monthly data, all the days will be 1 by default.

`Slope_cat` – Category of the current lifecycle stage, determined by Novartis through the combination of AI and business consensus. It can be: positive – growing trend -, neutral – stable trend - or negative – declining trend -. For a particular Country-Cluster, a change in the slope_cat means there is a change in the Lifecycle stage.

`Volume` – Analogue to sales, computed by: Volume = Sales / Price (per drug per country). This will be the target variable to be forecasted.

`Business_unit` – Business unit, can be either ONE or TWO.

`Ther_area_fact` – Encoded therapeutic area of the drug.

`Prevalence_pct` – Ratio from -100 to 100 of number of patients taking the drug with respect to the overall population.

Figure 2: Feature Description

Additionally, we have tables that provide information on launch-date, competitor-entry-date, indication-date, and pattern-expired-date for most of the clusters.

3. Exploratory Data Analysis

The first part of our problem was to find the transition point to maturity if the cluster keeps growing before 2019/04/01. Before proceeding to modeling, we wanted to have an estimated picture of what an actual growth-to-maturity curve would look like.

What we found out from drawing random growth-to-maturity curves was that lots of the curves do not have an obvious life cycle pattern. The major factor that caused this issue is the gap between the start date and the launch date of the cluster. For most clusters, the earliest available data started in 2012. But parts of clusters had a launch date much earlier than the start date of data recording, and this property would cause the cluster to have inaccurate stage-name labels.

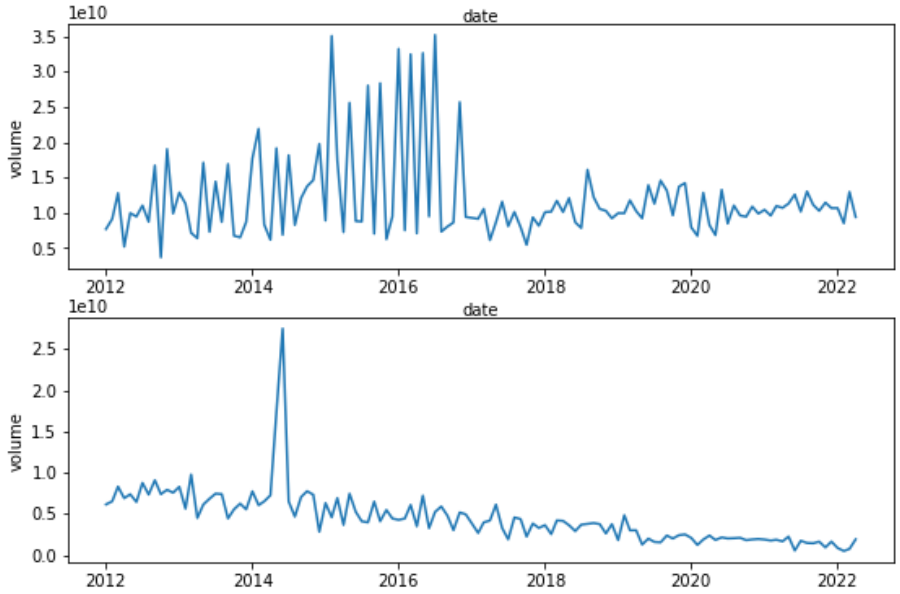


Figure 3: A peek into the clusters

We decided to look at the distribution of launch dates and came up with a threshold to filter out clusters with early launch dates. There were 422 gtm (growth-to-maturity) clusters with a launch date. Out of the 422 clusters, the number of gtm clusters before 2006 (including 2006) was 96, and the number of gtm clusters after 2006 was 326. We decided to use 2006 as the threshold to preserve as many clusters as possible while filtering out most of the noisy cases.

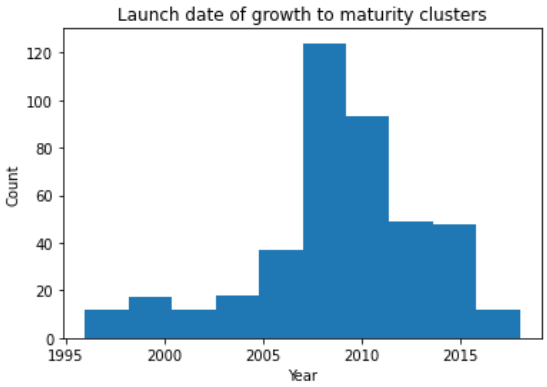
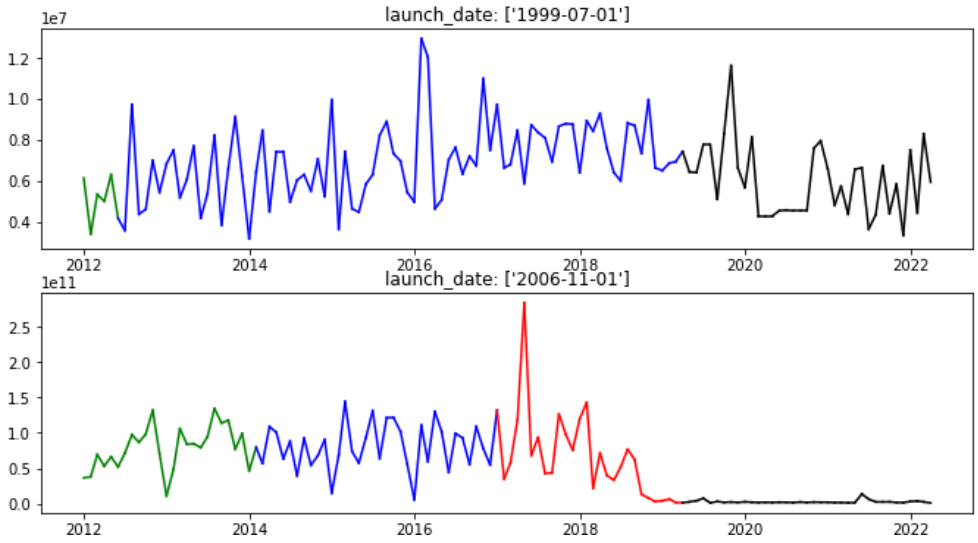


Figure 4: Distribution of the launch date

We drew some grow-to-maturity curves for clusters with launch dates before and after 2006 respectively. As expected, the grow-to-maturity curves with early launch dates tended to have a noisy trend, and the grow-to-maturity curves with close launch dates typically had a noticeable growth-to-maturity pattern. For the 326 clusters with launch dates after 2006, the average length of growth stages was around 56 months. We will use the average length of the growth stages to construct the baseline for the transition point prediction.



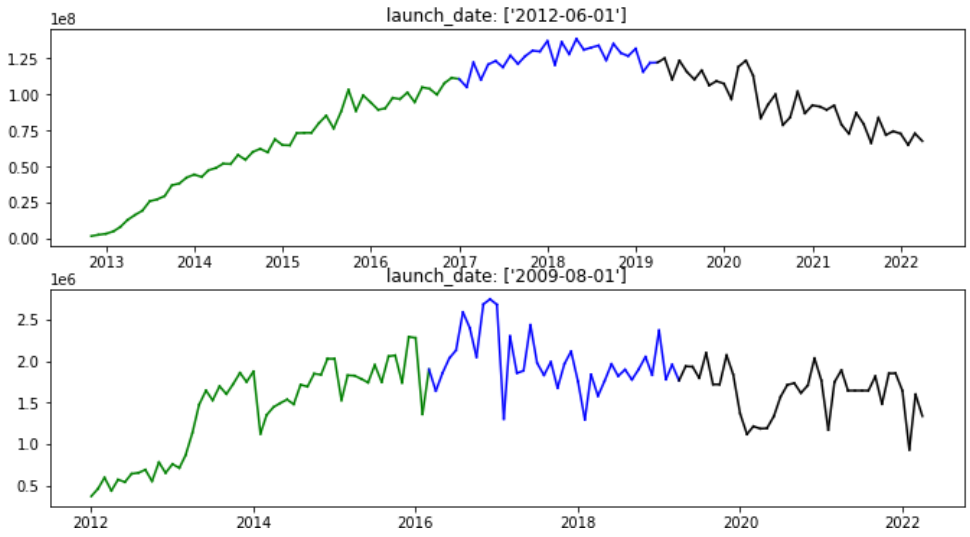


Figure 5: Example of gtm clusters (Green: Growth, Blue: Maturity, Red: Decline)

4. Preprocessing

4.1 Data Filtering

For data filtering, we filtered out clusters with launch dates before 2006, and only kept data with growth and maturity stage labels. We also used imputation to address the missing value issues in the auxiliary tables. For example, if the launch date was null, we would use the start date of the cluster as the launch date. There were also cases where the launch date was after the current date, and we set the start date for those clusters to the beginning of the cluster.

4.2 Feature Engineering

In order to extract information from available data, we brainstormed features that could potentially indicate transition. Then we performed feature engineering to come up with a final data frame for modeling.

We split the feature engineering among all group members, with each person in charge of the Python implementation of several features of his/her choice. The feature engineering Colab notebook is at Feature Engineering Notebook.

We then used an integrated function to perform the feature engineering step altogether. The input to the feature engineering function includes the data frames mentioned in the above table. After prefiltering to select from specific stages and times, we had 63,510 rows as the complete data. During the feature engineering step, we made sure that the number of data rows does not

change by avoiding joining from other tables directly, which will result in the increase of data amount. In other words, we made sure each data entry is indexed by (cluster_id, date). Each (cluster_id, date) combination will have one and only one entry in the final table we are working with. Upon the completion of feature engineering, we manually checked the correctness of the data by validating each feature and making sure the number of rows did not change.

The features implemented in this step include the following:

- **launch months**: how many months as of the current date since the launch date of that drug.
- **time_since_lost_exclusivity**: the number of months since the time this drug lost its exclusivity (if this drug never has exclusivity, return NaN).
- **transition_point**: if the transition point from growth to maturity happened (0 if not transition point, 1 if transition). Note this is the target variable of the problem we are working with.
- **transition point augmentation**: augment points surrounding the transition point by the given period.
- **volume_daily_avg**: daily average sales volume in the current month.
- **volume_lag_1m_pctg**: last month's daily volume/current daily volume.
- **volume_lag_2m_pctg**: second last month's daily volume/current daily volume.
- **volume_lag_3m_pctg**: third last month's daily volume/current daily volume.
- **vol_norm (from lag 1-6)**: normalized sales volume from the 6th last month to the last month.
- **vol_std_3m**: standard deviation of vol_nomr in the past 3 months.
- **vol_std_6m**: standard deviation of vol_nomr in the past 6 months.
- **vol_std_diff**: the difference between vol_std_3m and vol_std_6m.
- **Population**: the population of the country, note this feature does not take into account the monthly movement of the population but is only granular at the (country, year) level.
- **Reg_deg**: binary variable indicating the regulation designation state of the cluster.
- **comp_num**: the current number of competitors in the market, 0 if no competitors exist.
- **first_comp_month**: how many months since the first competitor entered the market (duration, 0 if no competitors, 1 for the first month of competitors entering).
- **if_exclusive**: binary variable indicating if at this date the drug has exclusivity.
- **month_since_growth**: calculate the number of months since the first growth stage was recorded.
- **change_rate**: we converted the volumes (number of sales) to change rate using the logarithm method so that we can solve the problem that different products have very different numbers of sales.

After the feature engineering step, we ended up with a data frame of 28 columns and 63,510 rows as expected. We also included the feature engineering quality check code in the notebook to make sure all of the features calculated run correctly as expected. In addition to checking the specific clusters, we also checked the overall distribution of some key features that can potentially bring us interpretable insights as follows:

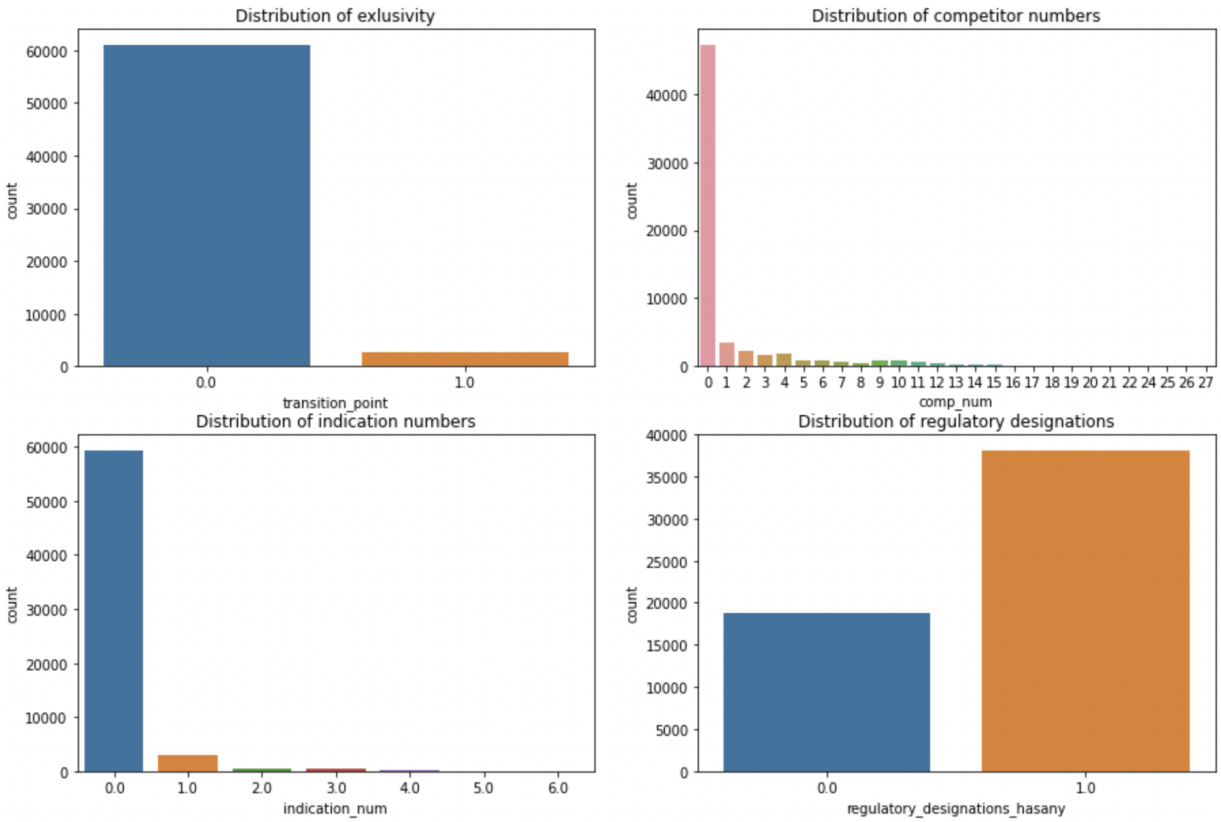


Figure 6: Distribution of four features including the target feature

From Figure 6, we can see that there exists some degree of imbalance in our data, which can potentially jeopardize the modeling performance.

5. Methodology

5.1 Train and Validation Split

The test set we obtained are time series dates between 2019/04/01 to 2022/04/01, and we will train our model on data before 2019/04/01 and make predictions on the test set. In light of this, we wanted to imitate the same testing process. We segmented our data into two intervals: before and after 2016. Clusters with a growth-maturity transition before 2016 were included in the training set; clusters without a growth-maturity transition before 2016 were included in the validation set. After the split, we had a total of 209 clusters in the training set and 349 clusters in the validation set.

5.2 Labeling

The target feature was whether a point is a transition point and we thought of two ways of labeling. The first was labeling the rows where the transition happens as 1 followed by augmentation, meaning that by default the previous and later 3 months will also be marked as a transition point. We did the augmentation mainly to tackle the problem of class imbalance. The second labeling method is using the provided growth and maturity labels and the problem turns into classifying whether a point is in the stage of growth and maturity. We then applied some algorithms to select the actual date for the transition. The labeling difference will also be later explained in our modeling section.

5.3 Evaluation Method

The standard accuracy metric was not suitable for our models as we can easily achieve high accuracy because of class imbalance. To gain valid insight into our models' performance, we chose to calculate the average difference between predicted and actual transition dates.

$$MAE = \frac{1}{n} \times \Sigma |actual\ date - predicted\ date|$$

Figure 7: MAE definition

We also identified two scenarios that will result in full percentage error, which are: when there was a transition point and our model predicted none; when there wasn't a transition point and our model predicted one.

5.4 Predicted Transition Point Selection

In both cases of labeling techniques, we also needed to select the transition point as there can be more than one point classified as 1 or maturity given a single cluster. We chose to either pick the first predicted transition point or the average predicted transition point. Also, in the case of 0/1 labeling, we tried to pick a threshold on how many transition points we need to see before we are confident that there is a transition. The intuition here was that since we have augmented the label for 6 more months, it could be a sign of no transition if few transition points are observed.

5.5 Volume Forecast

In the second stage, we chose change rate as our target feature. After training a model to predict the change rates, we use the benchmark value to convert the predicted change rate into actual volume prediction.

6. Models

6.1 First stage

In our first stage, we used a Random Forest Classifier and a Deep Neural Network Classifier to predict lifecycle stage transitions.

6.1.1 Baseline Performance

When we used the average length of the growth period before maturity as our predicting method, the average length of the growth period was 78.56 months. It would create an MAE of 16.40. When we don't make any predictions, the MAE in the validation set is 15.49.

6.1.2 Random Forest Classifier

The first model we chose was a random forest classifier for its capacity in handling binary classifications. For this model, we chose the 0/1 labeling technique. After catering all the features we needed, we constructed a pipeline including a one-hot encoder for categorical features, a standard scaler for numerical features, and a GridSearchCV object to search through three random forest parameters: max features, number of estimators, and class weight.

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('onehotencoder',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['country', 'cluster',
                                                   'cluster_id',
                                                   'business_unit',
                                                   'ther_area_fact',
                                                   'is_exclusive', 'month',
                                                   'regulatory_designations_hasany']),
                                                 ('standardscaler',
                                                  StandardScaler(),
                                                  ['indication_num', 'comp_num',
                                                   'first_comp_month',
                                                   'launch_months',
                                                   'month_since_growth', 'x1',
                                                   'x2', 'x3', 'x4'])])),
                ('randomforestclassifier',
                 RandomForestClassifier(class_weight={0: 1, 1: 20},
                                        max_features=10, n_estimators=200,
                                        oob_score=True, warm_start=True))])
```

Figure 8: Random Forest structure

From the search, our best model had the following parameters. We also achieved an 83.8 accuracy rate and 0.10 F1 score for this model.

```
('randomforestclassifier',
 RandomForestClassifier(class_weight={0: 1, 1: 20},
                        max_features=10, n_estimators=200,
                        oob_score=True, warm_start=True))])
```

Figure 9: Best Random Forest model parameters

We then applied the model to the validation set and calculated the errors. The mean absolute error of the model was around 18.37 months. We observed that a lot of errors came from the two scenarios we mentioned in the previous section. Moreover, in the case where the predicted and transition date both exist, the accuracy was not satisfying. We will discuss some limitations and future improvement directions in the next section.

6.1.3 Deep Neural Network Classifier

The Deep Neural Network model used stage names as the target label. Instead of predicting the transition point, it predicted if the stage was ‘Maturity’ or ‘Growth’. One issue we found when using transition point as the target label is that there was not much data with transition point labels even after augmentation. When using stage names as the label, we have way more data points with the Maturity label compared to the Transition Point label. Therefore, this model aimed to correctly predict if a given data point is at the Maturity stage or Growth stage. Figure 10 shows the Model summary.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                1152

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 2)                 66

=================================================================
Total params: 3,298
Trainable params: 3,298
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: DNN Model Summary

The Model that was selected to serve this purpose is a simple-to-implement DNN model with 2 hidden layers. The main reason to implement the DNN model is its simplicity to train, and the minimum required feature engineering.

As a result, the DNN model predicting Maturity or Growth has a bad performance compared to tree-based models. After getting the predictions, I picked the first date with a Maturity label as the predicted transition point. It reached an MAE of 23.97 months which is a -31.55% lift in performance. Besides, the model performance was very volatile. It can reach as good as 16

months or jump to 30 months. After tuning the model and training set, the stable model has an MAE of 24 months.

As a result, both models did not perform well in this task. The main reason was the provided features do not have a strong inference power on the transition. Secondly, the stage name itself might be inaccurate, as we can see some increasing and decreasing trends labeled as the ‘Maturity’ stage.

6.2 Second Stage

In our second stage, we trained multiple models to improve sales forecasting from the benchmark by damping the volumes after maturity. Two of them used transition points we got from the first stage and tried to tune the benchmark according to the predicted transition points while the other tried to forecast the sales directly without considering the transition point so that we can prevent bad performance due to inaccurate transition point predictions.

6.2.1 Train on Difference between Benchmark and Real Volumes

Before the transition points, drug sales were in the growth stage. So we simplified the trend by utilizing its positive slopes (slopes on change rates) and implemented linear extrapolation. We generated straight lines as our benchmark based on the historical data in the growth stages under assumptions that there were no transition points and all drug sales were in the growth stages. Then we were able to train regression models on differences between our benchmark and real sales volume to tune our benchmark.

Based on this method, we needed to do two steps: create the benchmark and train models to tune the benchmark.

6.2.1.1 Create Benchmark

Since we simplified the sales volume trend to a line, we first had to find the slope of the line for all training data. We came up with three ways to calculate the slope based on historical data. The first approach was to use the launch date point and the last point of the growth stage to calculate the slope. The second approach was to use the last point of the growth stage and one year ahead of the transition point to calculate the slope. The third approach was to use the last point of the growth stage and eighteen months ahead of the transition point to calculate the slope. Then, we calculated the sales volume based on the previously calculated slope. We evaluated these three methods by comparing the MAEs between the straight line and the actual sales volume. We found the third method had a slightly better MAE which was about $1.6 * 10^{11}$.

Figure 11 shows some examples of benchmarks, where the green lines are historical data, the blues lines are actual volume, and the red lines are calculated benchmarks based on the trend of the green lines.

```
Growth: Green
Mature: blue
Actural: balck
Predicted: red
```

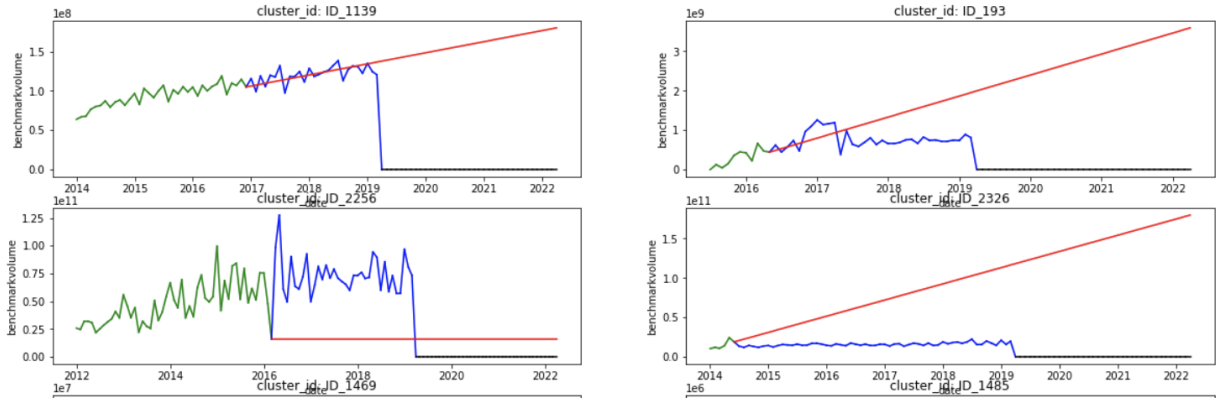


Figure 11: examples of benchmark

6.2.1.2 Tune Benchmark

With the benchmark generated, we tuned the benchmark prediction models to improve the performance. We included eighteen features to improve the difference between the actual volume and the benchmark prediction: indication_num, comp_num, first_comp_month, launch_months, month_since_growth, month_since_GtoM_trans, prevalence, time_since_no_exclusivity, population, country, cluster, cluster_id, business_unit, ther_area_fact, is_exclusive, month, regulatory_designations_hasany, year.

We have tested a linear regression, a gradient boosting regressor, and a random forest regressor on the training dataset we created. We evaluated the performance of the model using MAE, MSE, and R-squared, and concluded that the random forest regression model performs the best.

In order to achieve optimal performance on the benchmark prediction model, we utilized Sklearn's randomizedsearchCV and tuned for the following hyperparameters for the random forest regressor model: n_estimators, min_samples_split, min_samples_leaf, max_features, max_depth, bootstrap. After 3-fold cross-validation of 100 iterations, we have concluded the best parameters are:

- N_estimators: 1000
- Min_samples_split: 2
- Min_samples_leaf: 1
- Max_features: 'sqrt'
- Max_depth: 30
- Bootstrap: False

After the fine tuning, we were able to achieve 0.99 R-squared, $1.5 * 10^{21}$ MSE and $1.1 * 10^{10}$ MAE in out-sample tests.

6.2.2 Random Forest Regression Model Using Predicted Transition Points

This model avoids the complicated benchmark generating process. Instead, we filtered data in the training set and asked the model to learn the pattern during the ‘Maturity’ stage. After that, once a transition point is predicted, we would use this Random Forest Regression model to predict the corresponding ‘Maturity’ volume.

In order to improve the model performance, we did several attempts. The first thing we did was to transform the label ‘volume’(sales number) into a change rate, as different drugs have different magnitudes of sales numbers. Secondly, we excluded some drugs with very high volatilities from the training set, as we expect the ‘Maturity’ stage should have a stable sales number. Then, we re-evaluated the selected features to improve model performance, as including too many features would introduce uncertainty in some edge cases.

The Random Forest Model summary and the selected features are shown below:

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('onehotencoder',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['country', 'cluster',
                                                   'business_unit',
                                                   'ther_area_fact',
                                                   'is_exclusive', 'month',
                                                   'regulatory_designations_hasany',
                                                   'year']),
                                                 ('standardscaler',
                                                  StandardScaler(),
                                                  ['indication_num', 'comp_num',
                                                   'first_comp_month',
                                                   'launch_months',
                                                   'month_since_growth',
                                                   'month_since_GtoM_trans'])])),
                ('randomforestregressor',
                 RandomForestRegressor(n_estimators=50, n_jobs=-1,
                                       oob_score=True, random_state=0,
                                       warm_start=True))])
```

After yielding the prediction, the last action we did was to add a restriction condition on the prediction. If the model prediction indicates a very high volatile ‘Maturity’ sale number, we would manually flatten and smooth it.

As a result, we achieved a 0.22% lift in MSE and a 0.53% lift in MAE compared to the benchmark with seasonality and a 0.25% lift in MSE and 0.40% lift in MAE compared to the benchmark we generated.

```
performance_summary(pred_sub, actual = actual, benchmark = benchmark_selection(generate = True))

using average of the actual volume to calculate percentage (1.01e+11)
MSE: 1.94e+22 (190.45%)
RMSE: 1.39e+11 (138.00%)
MAE: 2.27e+10 (22.54%)
pred_pct: 11.73 (Percentage of Prediction by Row Level)
pred_cluster_id_pct: 31.42 (Percentage of Prediction by "cluster_id" Level)
MSE lift: 0.25%. RMSE lift: 0.12%. MAE lift: 0.40%.
```

```
performance_summary(pred_sub, actual = actual, benchmark = benchmark_selection(generate = False))

using average of the actual volume to calculate percentage (1.01e+11)
MSE: 3.86e+22 (379.50%)
RMSE: 1.97e+11 (194.81%)
MAE: 2.38e+10 (23.58%)
pred_pct: 11.73 (Percentage of Prediction by Row Level)
pred_cluster_id_pct: 31.42 (Percentage of Prediction by "cluster_id" Level)
MSE lift: 0.22%. RMSE lift: 0.11%. MAE lift: 0.53%.
```

6.2.3 Recurrent Neural Network Model without Knowing Transition Points

Because our stage-1 model did not have a strong performance, and its error can propagate to the second stage, we decided to experiment using LSTM models on the time series directly without using the transition point predictions.

We used all of the growth and growth-to-maturity clusters for training. There are 11 features for each month, and we were using 15 months to predict the change rate for the next month. We started by experimenting with different LSTM models, including vanilla LSTM, bidirectional LSTM, and LSTM with sequence-loss training. Bidirectional LSTM would process the input from both directions and concatenate the information in the hidden layer. With sequence-loss training, each hidden layer of LSTM corresponding to each month would make a prediction and compare with the next-month output, instead of just training the last hidden layer.

We started with small models and ran each setting using 5 random seeds, so the model initialization and train-val split would differ each time. The table below shows our initial results. In the MSE lift column of the first row, we have 35.52%(11.84%). The number 35.52% is the percentage of MSE performance increase compared to the benchmark, and 11.84% is the variance.

Model	Input length	Output length	Additional set up	Data	MSE lift	MAE lift
LSTM	15	1	hidden 15, 2 layers	all of growth	35.52%(11.84%)	-1.98%(1.7%)
LSTM	15	1	hidden 15, bidirectional,2 layers	all of growth	7.63% (17.74%)	-7.46% (0.96%)
LSTM	15	1	hidden 15, sequence_loss, 2 layers	all of growth	-46.06% (124.14%)	-27.46% (11.84%)

Table 1: initial LSTM experiments using 5 random seeds

These models had a very high variance, so our next step is to stabilize the model. We added three modules to our LSTM model: weighted loss, variance filter, and benchmark ceiling. The weighted loss module scales the loss by the size of the mean volume of each cluster, so the cluster with a higher average volume tends to have a higher loss. This module had the potential benefit of lowering the MSE. The variance filter helped us to get rid of the high variance clusters, and we found that taking out the top 5% of high-variance clusters works the best. By using the benchmark ceiling, we constrained the volume predictions to be not higher than the benchmark.

Model	Input length	Output length	Additional set up	Data	MSE lift	MAE lift
LSTM	15	1	hidden 15, bidirectional, 2 layers, weighted loss, variance filter, benchmark ceiling	all of growth	42.3%(0.001%)	0.17%(0.09%)

Table 2: the result of stabilized LSTM using 5 random seeds

Next, we performed an ablation analysis to investigate which module contributes the most to the stabilization of the model. We found that the variance filter module is the main contributor. Without it, the models start to fluctuate again. And the weighted loss module did not have a significant effect on the performance of our models.

Model	Input length	Output length	Additional set up	Data	MSE lift	MAE lift
LSTM	15	1	Without variance filter	all of growth	35.84%(8.92%)	6.37%(1.36%)
LSTM	15	1	Without weighted loss	all of growth	42.19%(0.004%)	-0.25% (0.29%)

Table 3: ablation analysis using 5 random seeds

Last, we tried to tune our model hyperparameters and experimented with different input lengths and settings. The number of hidden units was chosen from {15, 32, 64}, the number of LSTM layers is chosen from {1, 2, 3, 4}, and the number of final classification layers was chosen from {1, 2, 3}. We found that using input lengths of 10 or 25 would bring down the performance, so we stick with a length of 15. We experimented with using dropouts of 0.1, 0.3, and 0.5. And we also tried adding attention modules but it does not give better performance. Below is the final result of our tuned model using 10 random seeds. The result in the first row is our stabilized model, which gives a very persistent MSE performance increase but has a slightly negative MAE performance compared to the benchmark. The unstabilized version gives a positive performance for both MAE and MSE but has a higher variance.

Model	Input length	Output length	Additional set up	Data	MSE lift	MAE lift
LSTM	15	1	bidirectional,3 layers, hidden 32, 2 layers of fc, weighted loss, variance filter, benchmark ceiling	all of growth	42.23%(0.003%)	-0.54%(0.25%)
LSTM	15	1	bidirectional,3 layers, hidden 32, 2 layers of fc, weighted loss, benchmark ceiling	all of growth	35.61%(11.63%)	7.14%(2.09%)

Table 4: final results using 10 random seeds

7. Results

In this section, we compare the performance of different approaches we tried.

7.1 Transition Point Prediction

For the task to predict transition points, we compared the performance of the Random Forest Classifier and the DNN using mean absolute errors (MAE) and accuracy.

Model	MAE	accuracy
Baseline	16.4 months	0.4557
Random Forest	18.37 months	0.5103
DNN	23.97 months	0.4525

Table 5: performances for Baseline, Random Forest and DNN model

Though our Baseline has the smallest MAE, it has a lower accuracy than the Random Forest model. In reality, about half of the drugs don't have transition points so using the baseline seems unreasonable. Since the Random Forest model performs better, we used the predicted transition date generated by the Random Forest classifier for sales forecasting. Our poor performance on predicting the transaction points also led us to think of other ways to predict sales. For example, using the RNN model directly without the transition points is one of the methods we tried.

7.2 Sales Volume Forecasting

We evaluated three models for sales volume forecasting using their MAE liftings and MSE liftings compared to the given benchmark and generated benchmark.

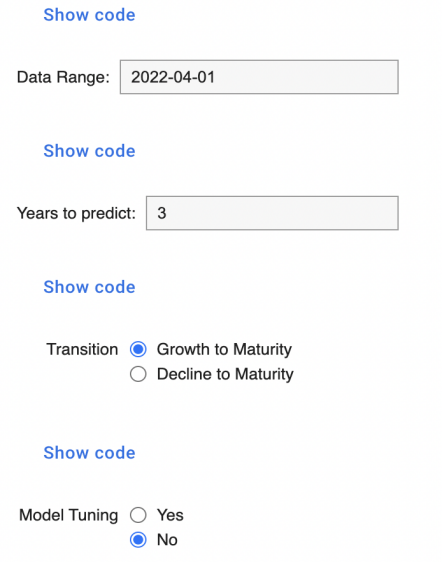
Model	MAE lifting (compared with Novartis benchmark)	MSE lifting (compared with Novartis benchmark)
Random Forest (using generated benchmark)	-2.00%	-0.55%
Random Forest (using Novartis benchmark)	0.53%	0.22%
Stabilized RNN	-0.54%	42.23%
Unstabilized RNN	7.14%	35.61%

Table 6: performance improvements for Random Forest and RNN model

For the Random Forest models, their improvements are smaller than 1%, but for the RNN model, we found a huge improvement in performance. This result matches our expectation, as we cannot predict the transition points accurately using the given data. Therefore, we can conclude that predicting transition points does not help much in predicting sales.

8. Pipeline

Our final deliverable is an end-to-end pipeline notebook that includes data preprocessing, feature engineering, and modeling. Moreover, it is a generalization of our work, granting users the ability to choose different time periods for training and testing and different transition types. Users can simply configure the following parameters through graphical buttons and text boxes:



- **Data Range**: This parameter specifies the data you want to include in the pipeline. The format needs to be exactly YYYY-MM-01. For example, the default is 2022-04-01, this means all data before 2022-04-01 will be included.
- **Years to Predict:** This parameter controls the split of training/testing data and it requires an output of integer with a unit of years. For example, the default is 3, this means data from 2019-04-01 will be the testing set, and data before that will be the training set.
- **Transition**: This parameter specifies what kind of transition we want to train on, the options are growth-to-maturity and decline-to-maturity.
- **Model Tuning**: This parameter decides if you want to tune the model trained. If yes, we will run a grid search. If not, we will use pre-specified parameters.

After the configuration, they can simply use the **run all** option to complete the whole process. Through the pipeline, both models mentioned above will be trained and saved for future access. The final output of the pipeline will be three CSV files and models.

9. Ethical Discussion & Future Work

In the process of this project, we started from scratch on understanding, analyzing, and modeling a business scenario that is typical in the operations of pharmaceutical companies: the identification of the market phase and the subsequent volume forecasting. Being able to understand different market phases is important to the business decision making, in that for markets in different stages, the company has different goals and therefore promotes different

marketing, developing, and pricing and strategies. In order to perform this task, we have experimented with multiple methods, including time series analysis, machine learning models, and deep learning models. The final model we proposed is a combination of multiple models that correspondingly serve its own purpose, including market phase identification, and volume regression given market-specific features. Also, we want to point out that the data given to us is already anonymized, hence there is no concern of information leakage for our data.

If given more time and resources, there are a few aspects of this project that can further improve its functionality in providing valuable insights to the pharmaceutical businesses. Firstly, we are not only interested in identifying the transition between growth and mature stages, but also other stages as well, especially the decline stage. Being able to identify all stages will complete the usability of the modeling framework and help the company make targeted investments and strategies with respect to different drugs and markets. Additionally, the volume forecast can be improved by adding more market-specific and drug-specific features to make the prediction values more robust and versatile. An observation we've made in evaluating the forecasts is that there still exist several outliers, which reflects the bias of our models on certain markets and drugs. Therefore, a way we can improve the current model is to look at the outliers and its projected country and drug to understand the sources of these biases, and then we can add the corresponding features to the model to improve the performance.

This project has been an opportunity to practice the methods we have learned at the M.S. in Data Science program at Columbia University, including but not limited to exploratory analysis, statistical analysis, software engineering, machine learning, and deep learning. In addition to taking these techniques into practice, we have also benefited from working with each other toward a common goal. In our team, each of us has contributed our own expertise into delivering this project, which will be helpful to our future career development.

10. Contribution

- **Hang Xu:** Managed team progress, performed feature engineering, designed the DNN classification model, constructed performance evaluation methods, helped fine tuning the regression model, and built the pipeline notebook.
- **Senqi Zhang:** Conducted EDA, performed feature engineering, designed the random forest models for both stages, and built the pipeline notebook.
- **Yajie Zhang**: Performed feature engineering and data quality checks, including visualization of some most influential features included in the feature engineering section (Fig. 5).
- **Shuyue Xu:** Conducted data cleaning and preprocessing, performed feature engineering and data quality checks, custom benchmark generating functions

- **Zehui Wu:** Conducted EDA, designed the algorithm for training/validation split and custom error calculation function, conducted RNN experiments.

11. References

[1] Connelly, F. J., & Daignault, G. (n.d.). *The life cycle concept as a long term forecasting model - journal of the Academy of Marketing Science*. SpringerLink. Retrieved December 11, 2022, from https://link.springer.com/article/10.1007/BF02729389