Customer Churn Prediction using Multi Time Slice Forecasting and Logit Leaf Model

By

Bokka Sai Pradeep

17QE30001

Under the supervision of

Prof. Jitendra Kumar Jha

Department of Industrial & Systems Engineering
Indian Institute of Technology Kharagpur
April 2022

Department of Industrial and Systems Engineering
Indian Institute of Technology Kharagpur
Kharagpur, India-721302

CERTIFICATE

This is to certify that this project entitled **Customer Churn Prediction using Multi Time Slice Forecasting and Logit Leaf Model** submitted by **Bokka Sai Pradeep (17QE30001)** to the Indian Institute of Technology Kharagpur, is a record of the bona fide project work carried out under my supervision and is worthy of consideration for award of Dual Degree of the Institute.

Prof. Jitendra Kumar Jha
Department of Industrial and Systems Engineering
Indian Institute of Technology Kharagpur
India - 721302

Date:

ABSTRACT

In an era of increasingly saturated industries and growing competition amongst enterprises, customer churn is a serious issue. As a result, companies and executives have realised that historical customer data from existing customer bases, which can be used to build models, is one of the most significant assets in the fight against customer churn. Customer churn prediction, as well as the search for and identification of customers with a high inclination to leave the company, are crucial.

So, here we have introduced a Machine Learning framework that contains training the dataset on multiple time slices to overcome the issues like the rarity of churn samples and the risk of overfitting. Then, followed by a Predictive Model named Logit Leaf Model(LLM) which is an implementation of Decision Trees followed by Logistic Regression, thereby compensating the drawbacks of the respective Algorithms.

Keywords: Customer churn prediction, Logit leaf model(LLM), Predictive Analytics, Operation Research in marketing, Hybrid algorithm

Contents

1. Introduction:

1.1 The Theme

Customer churn, also known as customer attrition, is when a company's customers stop buying or interacting with it. A high churn rate indicates that a large proportion of customers are no longer interested in purchasing goods and services from the company. The cost of keeping an existing customer is much lower than the expense of gaining a new one. As a result, it is a critical factor from a business standpoint.

A scoring model for customer churn prediction allows for the estimation of a future churn probability for each customer based on the customer's historical knowledge. In practice, these scores can be used to identify customers who should be targeted for a retention campaign. Previous research has looked at customer churn from two different perspectives. On the one hand, researchers are working to improve customer churn prediction models by developing and proposing more complex models to improve predictive performance. Researchers, on the other hand, want to know what causes customer churn and define key drivers of customer churn, such as customer satisfaction.

1.2 Problem Description

In a wide range of businesses, machine learning classifying models have been used to anticipate churn. Telecom, banking, and TV/newspaper subscription are all well-known examples of such industries. Models learn to identify customers who are likely to remain with the organisation and those who are likely to churn by training on a large group of potential customers.

The majority of churn prediction research uses a single time frame of data to train and test their models. By doing so, organizations possibly miss out on significant information from the data and fail to capture how churn conditions and their drivers vary over time. It's challenging to train a model that will maintain its performance in the future due to changing conditions. Predictive performance and comprehensibility are the two most important features for customer churn prediction, yet no known algorithm provides both.

Algorithms like Logistic Regression, Decision Tree, and Random Forest have all demonstrated good predicting capabilities.

So, here we have introduced an ML pipeline that can diminish the above-mentioned issues and can provide both good predictive performance and also good comprehensibility.

2. Literature Review:

2.1 Multi Time Slice Forecasting

Only a few studies consider multiple time periods within the training data. One exception is Gür Ali and Artürk (2014). Instead of the common approach of one sample per customer observed at one fixed point in time, they propose a sampling pipeline that uses multiple samples per customer observed at different points in time. The pipeline is being used in the banking industry, and the authors demonstrate that using multiple periods of training data improves predictive performance over only using one period of training data. The analysis, however, is limited to one example with a specified number of periods. The effect of varying the training set size and the number of included periods on predictive performance is not investigated.

The multi-slicing approach has shown promise in specific applications, but it has not been thoroughly studied, so the role of the training set size and the number of time slices, for instance, is unclear. It has yet to be determined whether performance improvement can be maintained when the training set size is orthogonal to the number of time slices, but samples are drawn from multiple time slices. This is important for comprehending the factors that influence performance improvement. It's also unclear how the number of time slices used for training impacts predictive performance.

Different types of Forecasting methods:

1. *Resampling:*

   Resampling is the process of expressing data at a different frequency. Let's pretend we have a temperature sensor that records data every second. We can use the average of 60 second observations in an minute and show the temperature changes every minute if we don't need second-level precision. This is known as downsampling, or converting to a lower frequency.

2. *Time Shift Forecasting:*

   Time-series data analysis may require to shift data points to make a comparison. The **shift** and **tshift** functions shift data in time.

- shift: shifts the data
- tshift: shifts the time index

3. *Max-Min Forecasting:*

   It considers the maximum or minimum of values in a particular interval of time(ex: 1 hour or 1 month).

The above mentioned Forecasting methods consider only a single time slice of data or does not include the past data. Whereas Multi Time Slice Forecasting considers multiple time slices of data which in turn increases the correlation of the present data on the past data that is more helpful in churn prediction.

2.2 Logit Leaf Model

By consequence customer churn prediction is a complex process that requires informed decisions from an analyst or researcher at various stages (Lima, Mues, & Baesens, 2011). Lessmann and Voß (2009) highlight the significance of data-driven decisions whenever possible. In the modelling stage, the main decision point for an analyst is which of the classification algorithms to use, with a trade-off between comprehensibility and predictive performance. This explains why DT and LR are popular churn prediction algorithms because they combine good predictive performance with comprehensibility.

The LLM is a two-stage hybrid technique that uses a decision tree to identify homogenous customer segments in the first step and then applies logistic regressions to each of these segments in the second step.

The LLM has gotten a single + for comprehensibility because of its hybrid nature and reduced simplicity. Nonetheless, the hybrid LLM algorithm has the ability to recognise segments, which has been shown to be useful in customer churn research due to the heterogeneity in the customer base, as well as to identify important customer churn reasons specific to those segments. As a result, it gives management access to useful and actionable insights that can help them set up segment-specific targeted retention strategies based on the major customer churn factors. As a result, in the context of customer churn prediction, the LLM may be recommended over the LR or DT.

Logit modelling has been found to produce reliable and satisfactory results in churn prediction benchmarking experiments, and it can compete with more advanced techniques. Forward selection is used in logistic regressions in the LLM, which means that the method (1) contains a built-in feature selection process and (2) selects the most essential variables for each group individually.

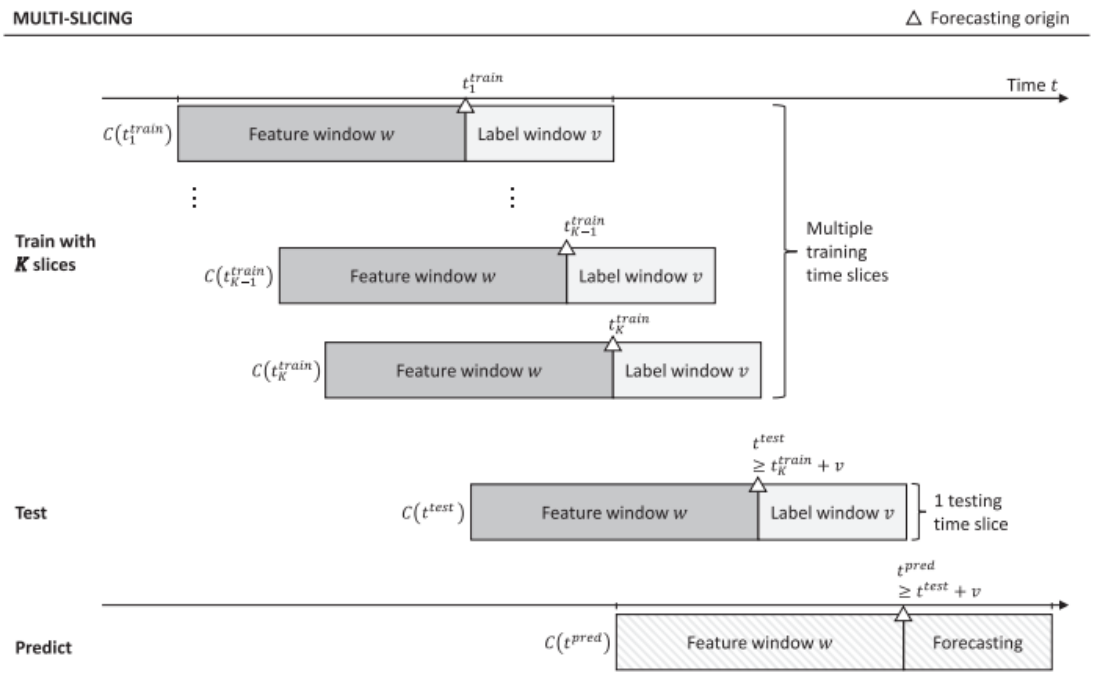
3. Methodology:

3.1 *Multi-Slicing*

For multi-slicing, transactional time-stamped data must be processed so that multiple time slices can be used for training.

A time slice is a portion of the data for a given time period that has been extracted. It has both feature and label windows, with the forecasting origin t as a reference point in between.
Multi-slicing is made possible by dividing accessible transactional data into time slices. Multiple time slices can be formed and integrated into a single training set using this idea. Customers are observed at different moments with multi-slicing. ttrain k ttrain k ttrain k ttrain Each time slice k 1,..., K has features and labels that represent a snapshot of the customers at the forecasting origin ttrain k. Customers at ttrain K are seen in the most recent slice, whereas customers at ttrain 1 are shown in the oldest slice, which has been pushed backwards by K-1 periods. A stacked feature matrix XK,ttrain = [Xttrain 1 Xttrain K] and a corresponding stacked label vector make up the training data set. There are up to k |C(ttrain k)| rows in XK,ttrain. Note that instead of using only one time slice, K-1 additional periods of data are used for training on K time slices.

Our method of multi-slicing is neither an under-sampling nor an over-sampling technique. Multi-slicing, on the other hand, allows more real observations from historical data from both the minority and majority classes to be accessed without affecting the underlying class distribution. While just moderately widening the time horizon used, creating time slices generates additional samples and addresses the absolute rarity or lack of data issue. As a result, we do not delete any data and instead make use of all available information. Multi-slicing allows you to learn about previous churners and non-churners as well as the most recent ones.



Training on multi time slices

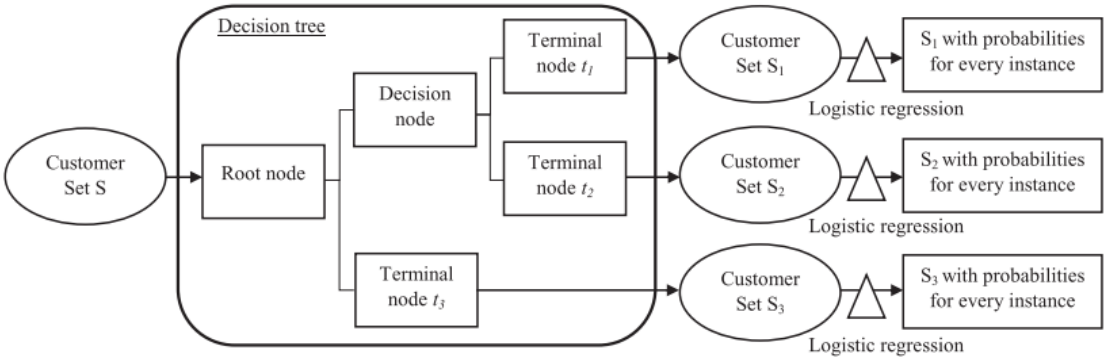
3.2 *Predictive Modeling using LLM*

Decision trees are a sort of prediction model that performs well in terms of efficiency and comprehensibility due to their simplicity. They use a strategy that involves executing an iterative search throughout the decision tree branches which are nothing but a set of data points and picking the best set of branches based on the Gini-index to recursively split the data into smaller and purer subsets. This method starts with the root node, which has no node pointing towards it and iteratively chooses the optimal splitting criterion for dividing it into 2 child nodes and so on. When no more splits are feasible, the process concludes with a set of nodes known as leaves, which are nodes without child nodes or simply there are no nodes pointing away from them.

Splitting data repeatedly results in more complicated models and, as a result, a higher risk of overfitting. On the one hand, this unwanted complexity may be decreased by applying pruning and establishing a set of tuning parameters that influence the iterative splitting procedure. Pruning is a method of lowering complexity that includes deleting tree parts that lack the ability to categorise customers. Several decision tree tunings, such as the minimum

leaf size, impact the tree's complexity by finding the least optimal number of splits in nodes and that serves as a halting criterion for this iterative dividing process.

In conventional DT, observations are assigned predictions based on the class trend of the leaf where they occur. Fitting forward feature selection logistic regression at each leaf or terminal node is the second stage of the LLM technique.



Conceptual representation of the Logit Leaf Model

3.3 *SMOTE Algorithm*

Synthetic Minority Oversampling Technique (SMOTE) is a method for expanding the number of data samples in a dataset proportionately. Based on the existing minority classes you provide as input, the module creates new instances. As a result of SMOTE implementation, the number of majority cases does not change.

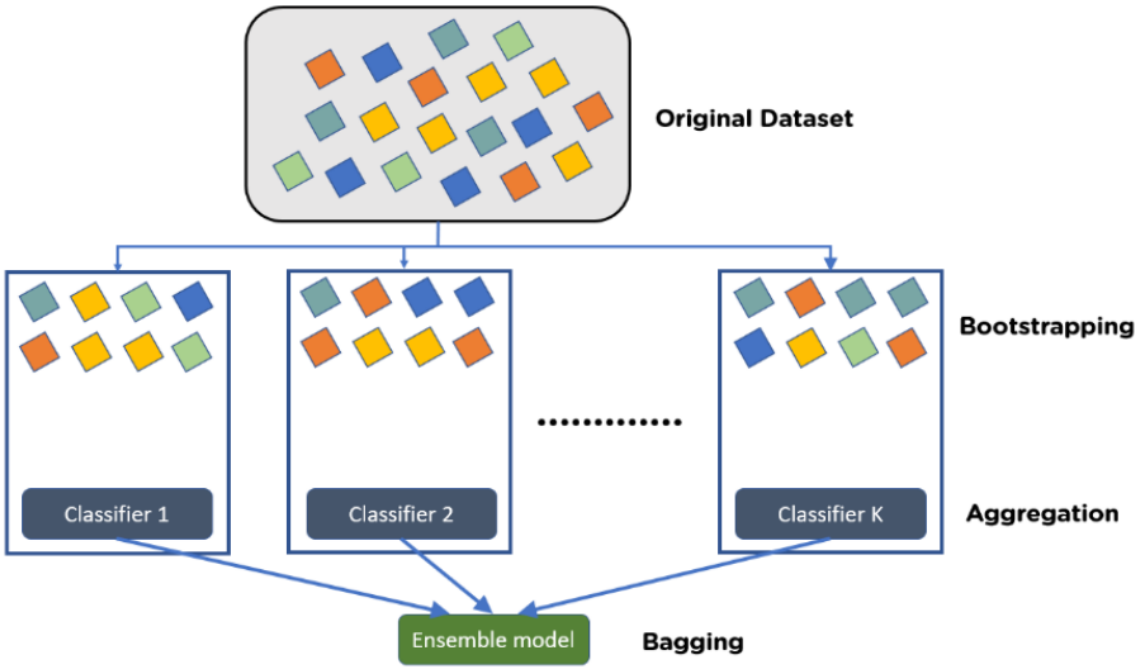
The new cases aren't merely duplicates of existing minorities. Instead, the method obtains feature space samples for each target class and its immediate neighbours. The algorithm then creates new instances that merge the target class's characteristics with those of its neighbours. This method increases the data samples available to each class in a 1:1 ratio and makes the data points more general.

We apply SMOTE technique to some leaves that occur as soon as after applying the Decision Tree(DT) Algorithm to the dataset since some of the leaves contain only imbalanced classes.

So, it then helps in generating new instances from the existing minority class. We then apply Logistic Regression(LR) to each leaf.

3.4 *Bagging*

Bagging, also known as Bootstrap aggregating, is an ensemble learning strategy that helps machine learning algorithms increase their performance and accuracy. It decreases the variance of a prediction model by dealing with bias-variance trade-offs. Bagging is a technique for avoiding data overfitting and is used in both regression and classification models, as well as decision tree techniques.



The Bagging technique ensures to give more generalised output which is more helpful incase of large datasets. The datasets for churn prediction are always large. Hence, we are trying to add bagging technique by creating an ensemble model with different lengths of decision trees.

We create ensemble model by taking the mode of the outputs obtained with different lengths of trees.

4. Experimental Setup:

4.1 Dataset

I have used the data of each cab driver's activity to be able to predict when a particular driver might churn. This gives the added advantage of being able to take actions accordingly and in a timely fashion. I have used a directory containing 9 datasets in which the data of every customer for a particular month is given in each dataset. Therefore, 9 datasets represent the data for 9 months(Jan - Sep).

Data Source: https://github.com/sayondutta/recurrent_neural_networks_to_predict_churn

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	user_id	net_last_b	location_c	home_grid	office_gric	requests_s	requests_a	travel_dist	travel_cou	free_rides	recency_m	points_ear	route_dist	user_gend	avg_rating	churn
2	0	487	0	0	0	0	0	0	0	0	20000	0	30	1	0	0
3	76	1780	0	0	0	0	0	18	2	0	3843.567	48	9	1	0	0
4	82	880.5	1	2	54	2	20	250	25	9	2204.75	882	10	1	4.777778	1
5	108	4830.4	1	1	54	1	1	31	1	0	26453.65	126	31	1	0	1
6	138	2290.5	0	7	131	11	30	693	33	0	7782.733	1683	21	1	4.9	1
7	145	52.5	0	0	0	0	0	22	1	0	17652.93	63	22	1	0	0
8	169	0	0	0	0	0	0	23	1	0	23515.87	0	23	0	0	0
9	210	58.5	0	2	5	13	1	10	1	0	36483.58	36	10	1	0	0
10	285	346.5	1	3	54	2	3	16	4	4	4308.95	0	4	0	5	1

A snippet of one of the datasets (Jan)

The directory of datasets contains the data of 42,458 cab drivers activity every month for 9 months.

4.2 *Experimental Design*

4.2.1 Data Preprocessing

1. The first step involves making a unique dataset which is occurred by merging all the individual datasets(i.e. 9).
2. Missing data imputation is used at first. Missing values are treated differently based on the amount of missing data in a particular feature. Imputation techniques are used for the features with missing data of more than 5%. Depending on the feature, zero, median and modus imputations are used. Dummy variables are used to detect features that require imputed missing variables. For characteristics with the missing data of less than 5%, the datapoint having the missing data are eliminated from the dataset to control the impact of imputation methods.
3. To convert category values to binary variables or features, dummy variable encoding is utilised. This method creates x–1 dummy features, where x is the number of various categorical variable values. These newly created features showcase the absence or presence of a specific feature. Here only x-1 features are created, so as to avoid dummy variable trap.
4. Outliers must be identified and treated in the fourth phase. Outliers are anomaly values that deviate by more than 3 standard deviations from the mean or average value of a variable. Though it has not much effect why applying the Decision Tree, but is helpful in the later step of applying Logistic Regression to individual leaves. Winsorization is used for converting outliers to "acceptable" values that are in the range of 3 standard deviations.

4.2.2 Data Modeling

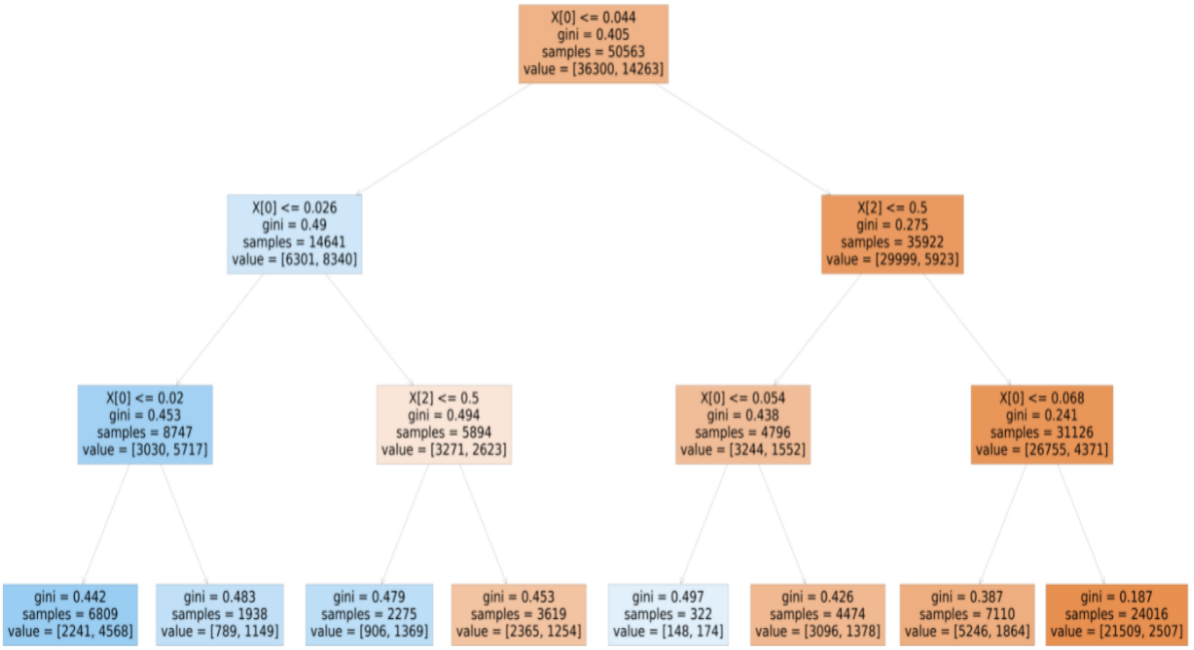
1. After the Preprocessing part is completed, the dataset is then trained on multiple time slices considering a time window of 3 months by taking the rolling mean of 3 consecutive samples of the dataset throughout the dataset by using the formula:

$$SMA_k = \frac{p_{n-k+1} + p_{n-k+2} \cdots + p_n}{k}$$
$$= \frac{1}{k} \sum_{i=n-k+1}^{n} p_i$$

2. Then the entire dataset is scaled so that all the values lie between 0-1 which helps Gradient descent to descend faster. Even though it has not much effect why applying the Decision Tree, but is helpful in the later step of applying Logistic Regression to individual leaves.
3. Before continuing to the next step, the whole dataset is divided into training and testing datasets in an 80:20 ratio.
4. The Decision Tree Algorithm is applied to the scaled dataset with pruning so as to obtain the optimal number of leaves.
5. The SMOTE Algorithm, a method for over-sampling is then applied on the leaves with an imbalance.
6. Later, Forward Selection Logistic Regression is applied on each leaf so as to select the most essential churn drivers for each group individually.
7. Steps 4-6 are repeated again for 3 different depths of decision trees i.e. 3,4,5.
8. The next step is Bagging i.e. Taking the mode of the predictions of each data sample obtained from each individual decision tree of different lengths.

5. Results:

i. Logit Leaf Model:



From the above snapshot, we can observe the resultant Decision Tree with 8 leaves out of which the leaves 7 and 8 have the label imbalance which has to be dealt with SMOTE Algorithm explicitly.

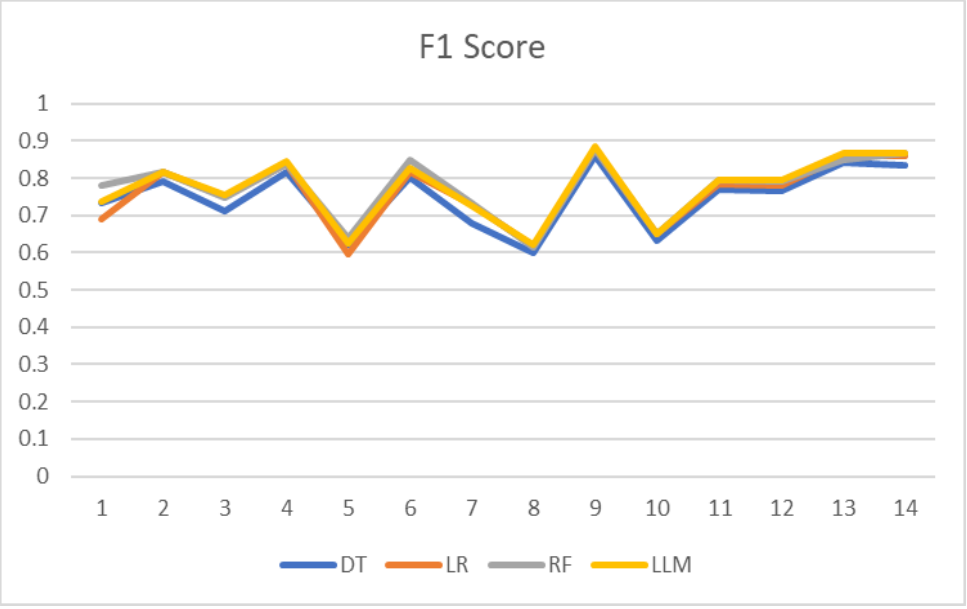
```
#Left half
L1=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]<=0.026) & (X.iloc[:,0]<=0.02)]
L2=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]<=0.026) & (X.iloc[:,0]>0.02)]

L3=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]>0.026) & (X.iloc[:,2]<=0.5)]
L4=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]>0.026) & (X.iloc[:,2]>0.5)]
#Right half
L5=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]<=0.5) & (X.iloc[:,0]<=0.054)]
L6=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]<=0.5) & (X.iloc[:,0]>0.054)]

L7=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]>0.5) & (X.iloc[:,0]<=0.068)]
L8=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]>0.5) & (X.iloc[:,0]>0.068)]
```

The above code represents the accessing of an individual leaf.

	Decision tree (DT)	Logistic regression (LR)	Random forests (RF)	Logit Leaf model (LLM)
Ds1	0.734 (0.013)	0.691 (0.004)	0.782 (0.008)	0.739 (0.011)
Ds2	0.792 (0.008)	0.816 (0.004)	0.816 (0.003)	0.816 (0.004)
Ds3	0.712 (0.018)	0.753 (0.012)	0.748 (0.010)	0.754 (0.012)
Ds4	0.816 (0.005)	0.840 (0.001)	0.839 (0.001)	0.846 (0.001)
Ds5	0.620 (0.006)	0.595 (0.003)	0.639 (0.004)	0.626 (0.004)
Ds6	0.804 (0.007)	0.815 (0.003)	0.848 (0.004)	0.827 (0.004)
Ds7	0.678 (0.018)	0.731 (0.009)	0.735 (0.012)	0.726 (0.011)
Ds8	0.601 (0.008)	0.618 (0.005)	0.614 (0.006)	0.620 (0.005)
Ds9	0.861 (0.007)	0.879 (0.002)	0.877 (0.002)	0.887 (0.002)
Ds10	0.634 (0.015)	0.651 (0.009)	0.649 (0.009)	0.652 (0.009)
Ds11	0.768 (0.006)	0.783 (0.002)	0.796 (0.002)	0.794 (0.002)
Ds12	0.766 (0.015)	0.781 (0.003)	0.793 (0.003)	0.794 (0.003)
Ds13	0.843 (0.007)	0.861 (0.002)	0.848 (0.003)	0.867 (0.002)
Ds14	0.835 (0.005)	0.861 (0.002)	0.868 (0.002)	0.867 (0.002)

Comparison of performances of different algorithms on churn prediction

The above is the comparison of the performance(F1 score) of several proven algorithms for each K-fold in which the ones with higher performance in each dataset is underlined. We can see that Logit Leaf Model has been underlined more times (7) indicating it to be the better classification algorithm of all. The values enclosed in the brackets indicates the p-value which tells us the probability that an observed difference could have occurred just by random chance(It has to be minimum as far as possible).

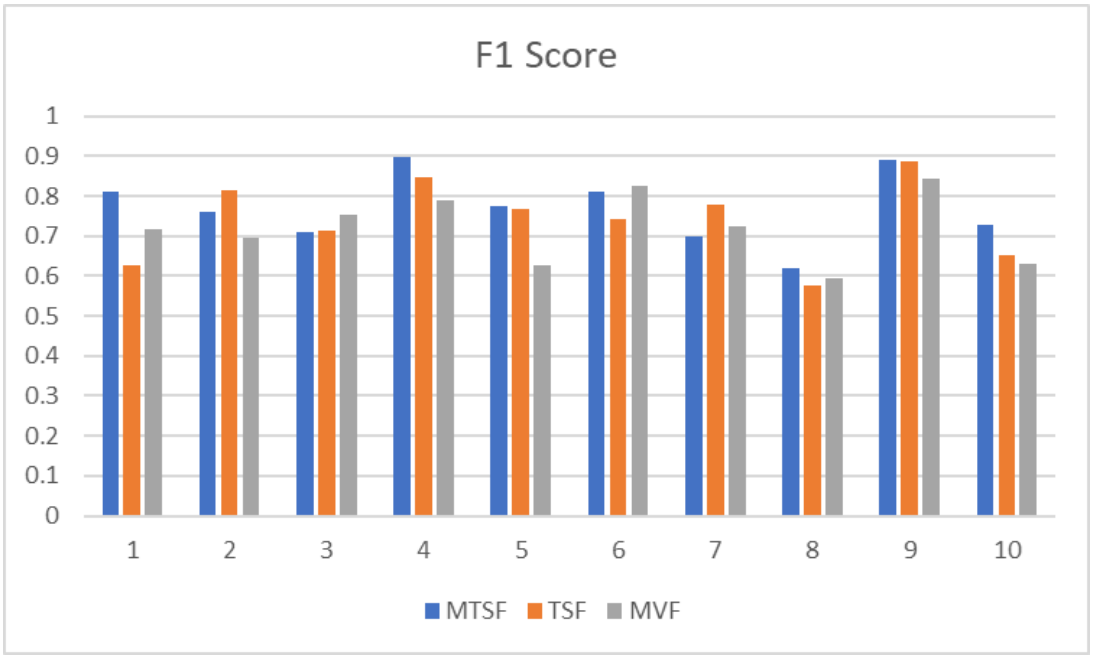


The plot of Comparison of performances of different algorithms

Multi Time slice Forecasting	Time Shift Forecasting	Max Value Forecasting
0.739	0.626	0.718
0.816	0.784	0.697
0.712	0.748	0.754
0.846	0.867	0.789
0.768	0.687	0.626
0.743	0.751	0.827
0.780	0.649	0.726
0.620	0.638	0.594
0.887	0.763	0.842
0.652	0.648	0.632

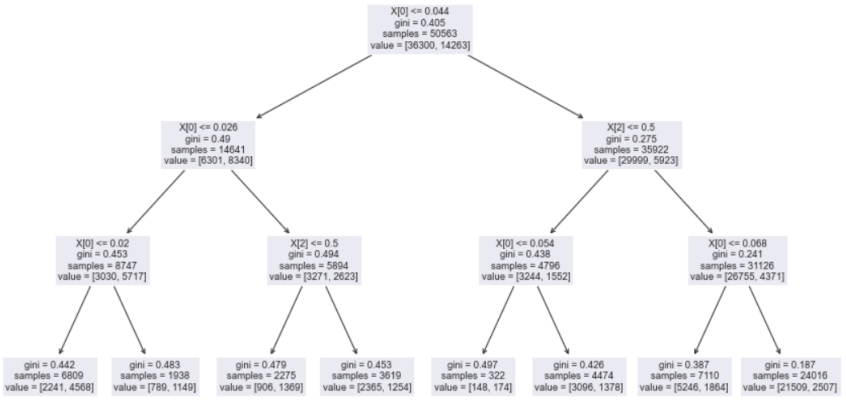
Comparison of different Forecasting techniques

It can be seen that Multi Time Slice Forecasting shows better performance than any other technique.

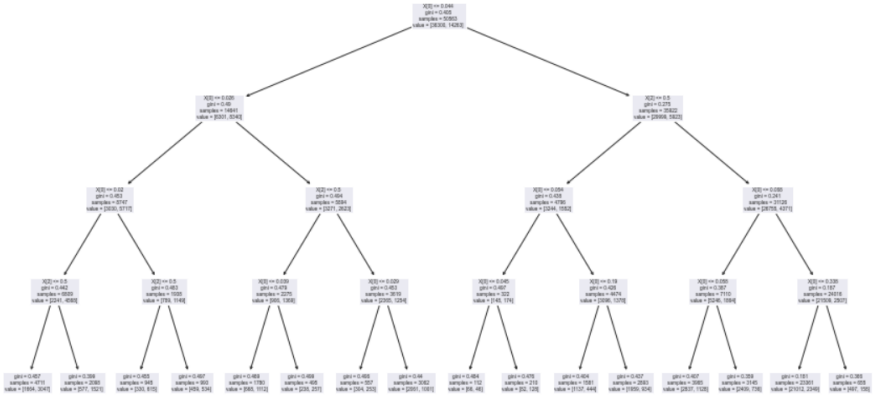


Comparison of performances of different Forecasting techniques on churn prediction

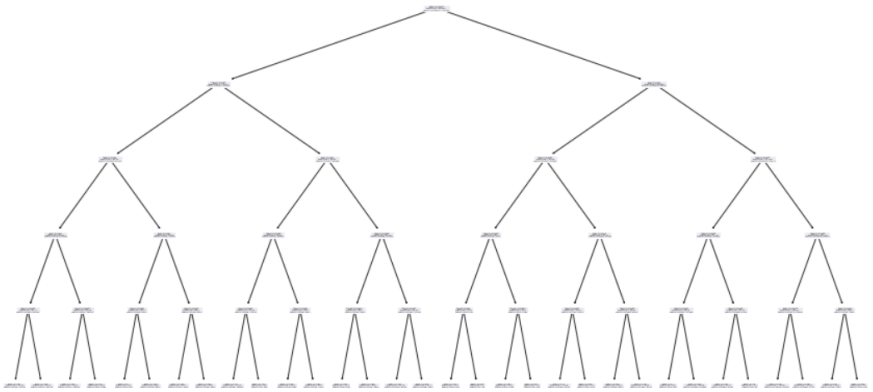
ii. Bagging:



Decision Tree with depth 3



Decision Tree with depth 4

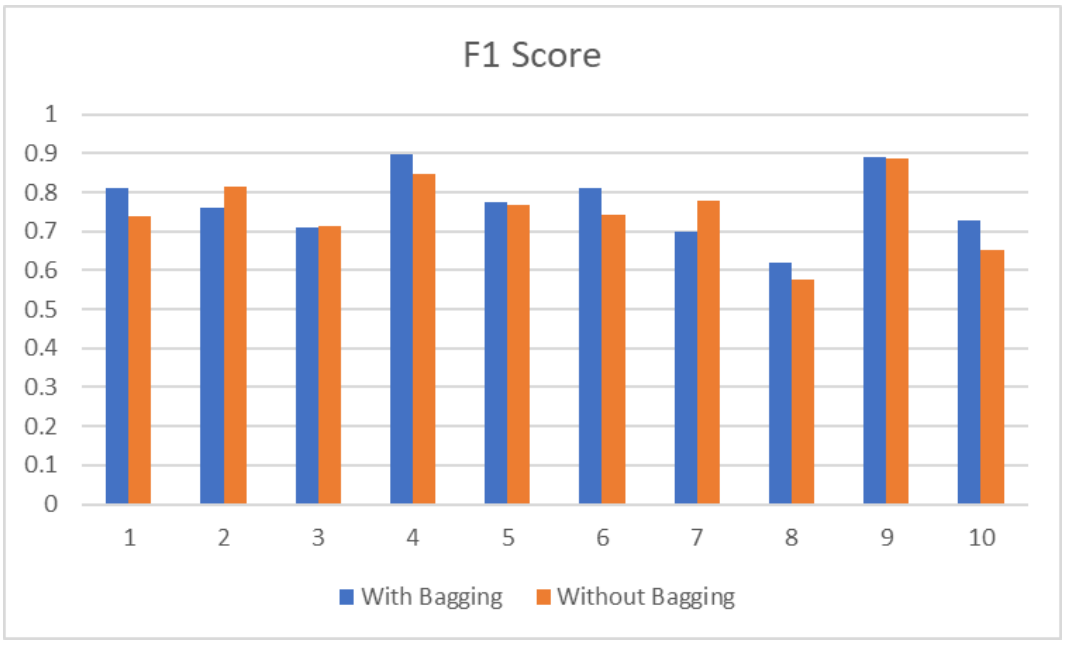


Decision Tree with depth 5

With Bagging	Without Bagging
0.811	0.739
0.761	0.816
0.711	0.712
0.897	0.846
0.776	0.768
0.813	0.743
0.698	0.780
0.620	0.575
0.892	0.887
0.727	0.652

Comparison of performances

We can see that the performance when bagging technique is improved than before bagging. It has been underlined 7 times.



Comparison of performance of the model with and without bagging

6. Limitations:

I felt it to be challenging during the testing phase. Since it's a completely new model, we couldn't use any pre-defined sklearn functions to obtain several metrics. So, I had to calculate the metrics explicitly for every data sample which is a much time taking process.

7. Conclusion:

We have introduced a pipeline with a new training method i.e. Multi Time Slice Forecasting and a new algorithm i.e. Logit Leaf Model. Typically, the choice of a classification technique is a trade-off between comprehensibility and predictive performance. Hence, LLM can be considered a reliable choice that scores well on both the essential requirements. As proven from the results, the LLM provides more high scoring models than using its building block algorithms, Logistic Regression and Decision Trees, as standalone classification methods.

The main intuitions behind the exceptional performance of this pipeline are:

1. The training phase of the model includes considering a time slice or window that includes training on multiple periods of data instead of only one which ensures not to lose the dependency of the past data on the present data.
2. The first part of the LLM algorithm i.e. Decision Trees performs the feature selection indirectly and divide the data into several groups. Whereas the second part involves selecting the important features for each group independently which takes care of the comprehensibility factor i.e. finding out the churn drivers also for each customer if the customer is predicted to get churn in the future.
3. The Bootstrap aggregating(Bagging) ensures that the prediction made is a more generalised one since it includes the ouput of many decisions tress of different lengths at the same time.

8. References:

1. Anderson, D. R., Sweeney, D. J., Williams, T. A., Freeman, J., & Shoesmith, E. (2010). Statistics for business and economics. Cengage: Andover (2nd ed.).

2. Athanassopoulos, A. D. (2000). Customer satisfaction cues to support market segmentation and explain switching behaviour. Journal of Business Research, 47(3), 191–207.

3. Baesens, B., Setiono, R., Mues, C., & Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. Management Science, 49, 312–329.

4. Ballings, M., & Van den Poel, D. (2012). Customer event history for churn prediction: How long is long enough? Expert Systems with Applications, 39(18), 13517–13522.

5. Bose, I., & Chen, X. (2009). Quantitative models for direct marketing: A review from a systems perspective. European Journal of Operational Research, 195(1), 1–16.

6. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

7. Chen Z.-Y., Fan Z.-P., Sun M. A hierarchical multiple kernel support vector machine for customer churn prediction using longitudinal behavioural data – European Journal of Operational Research.

8. Coussement K., Lessman S. & Verstraeten G. A comparative analysis of data preparation algorithms for customer churn prediction: A case study in telecommunication – Decision Support Systems.

9. Leung, H. C., & Chung, W. (2020). A Dynamic Classification Approach to Churn Prediction in Banking Industry. In Amcis 2020 proceedings data science and analytics for decision support.

10. Piatetsky-Shapiro, G., & Masand, B. (1999). Estimating campaign benefits and modelling lift. In Proceedings of the fifth ACM SIGKDDinternational conference on knowledge discovery and data mining - KDD '99 (pp. 185–193). New York, New York, USA: ACM Press.

11. Reichheld, F. F. (1996). Learning from customer defections. Harvard business review, 74(2), 56–69.

12. Tang L., Thomas L., Fletcher M., Pan J. & Marshall A. Assessing the impact of derived behaviour information on customer attrition in the financial service industry – European Journal of Operational Research 2014.

9. Appendix:

Key Codes:

```
for i in range(1,len(df)-1):
    if(df.iloc[i,0]==df.iloc[i+1,0] and df.iloc[i,0]==df.iloc[i-1,0]):
        df.iloc[i+1,1]=max(df.iloc[i,1],df.iloc[i-1,1],df.iloc[i+1,1])
        df.iloc[i+1,7]=max(df.iloc[i,7],df.iloc[i-1,7],df.iloc[i+1,7])
        df.iloc[i+1,9]=max(df.iloc[i,9],df.iloc[i-1,9],df.iloc[i+1,9])
        df.iloc[i+1,11]=max(df.iloc[i,11],df.iloc[i-1,11],df.iloc[i+1,11])
        df.iloc[i+1,13]=m(df.iloc[i,13],df.iloc[i-1,13],df.iloc[i+1,13])
```

Code representing the implementation of Multi Time Slice Forecasting

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf = DecisionTreeClassifier(max_depth = 3,random_state = 42)
clf.fit(X_train, y_train)

clf1 = DecisionTreeClassifier(max_depth = 4,random_state = 42)
clf1.fit(X_train, y_train)

clf2 = DecisionTreeClassifier(max_depth = 5,random_state = 42)
clf2.fit(X_train, y_train)
```

Code representing the creation of different decision trees of different lengths before the Bagging step

PseudoCode of LLM:

Model creation phase
Split D_{tot} in D_{tr} and D_{val}
Input: (training) data $D_{tr} = \{(Xi, Yi)\}_{i=1}^{N}$
1 : Calculate initial decision tree on D_{tr} spanning the total space S
2 : Define subspaces S_t based on a set of terminal nodes T for which $S = \bigcup_{t \in T} S_t, \forall\, t \neq t'$: $S_t \cap S_{t'} = \emptyset$
3: **For** $i = 1$ *to* T **do:**
4: Define starting logit
5: **Repeat**
6: add variable max decrease AIC
until stopping criteria
7: **End for;**
8: Combine results M_k in model M
Output: model M

Prediction phase
Input: (new) data $D_{val} = \{(Xi, Yi)\}_{i=1}^{N}$
1: Apply decision rules of model M on D_{val} spanning the total space S, resulting in subspace S_t based on a set of terminal nodes T for which $S = \bigcup_{t \in T} S_t,\ \forall\, t \neq t'$: $S_t \cap S_{t'} = \emptyset$
2: **For** $i = 1$ *to* T **do:**
3: Apply logistic regression specific for S_t
4: **For** $j = 1$ *to* n_i **do:**
5: Calculate predictions for all n_i instances in S_t
6: End For;
7: End For;
8: Combine predictions
Output: one prediction for every instance in S

```
#Left half
L1=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]<=0.026) & (X.iloc[:,0]<=0.02)]
L2=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]<=0.026) & (X.iloc[:,0]>0.02)]

L3=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]>0.026) & (X.iloc[:,2]<=0.5)]
L4=X.loc[(X.iloc[:,0]<=0.044) & (X.iloc[:,0]>0.026) & (X.iloc[:,2]>0.5)]
#Right half
L5=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]<=0.5) & (X.iloc[:,0]<=0.054)]
L6=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]<=0.5) & (X.iloc[:,0]>0.054)]

L7=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]>0.5) & (X.iloc[:,0]<=0.068)]
L8=X.loc[(X.iloc[:,0]>0.044) & (X.iloc[:,2]>0.5) & (X.iloc[:,0]>0.068)]
```

The above code represents the accessing of an individual leaf

```
from sklearn.linear_model import LogisticRegression
```

```
lr = LogisticRegression(solver='liblinear', random_state=0)
```

```
from sklearn.feature_selection import SequentialFeatureSelector as sfs
sfs = sfs(lr, n_features_to_select=1, scoring='f1')
sfs=sfs.fit(L8.drop(['churn'],axis=1),L8['churn'])
X=sfs.transform(L8.drop(['churn'],axis=1))
L8_red=pd.DataFrame(X)
```

```
sfs=sfs.fit(L3.drop(['churn'],axis=1),L3['churn'])
X=sfs.transform(L3.drop(['churn'],axis=1))
L3_red=pd.DataFrame(X)
L3_red
```

Code representing the implementation of Logistic Regression on one of the leaf after the Decision Tree step