


A New Churn Prediction Model Based on Deep Insight Features Transformation for Convolution Neural Network Architecture and Stacknet

Jalal Rabbah, Hassan 2 University, Morocco*

 <https://orcid.org/0000-0001-5615-7638>

Mohammed Ridouani, Hassan 2 University, Morocco

Larbi Hassouni, Hassan 2 University, Morocco

ABSTRACT

Predicting churn has become a critical issue for service providers around the world, in particular telecom operators for whom acquiring new customers is four times more costly than retaining existing ones. To keep up with the market, considerable investments are made to develop new anti-churn strategies, including machine learning models that are increasingly used in this field. In the work, the authors combine three stages. In first stage, by using deepInsight, they transform the attributes of dataset into images in order to take advantage of the strength of convolution networks in detecting hidden patterns in the dataset. In the second stage, they use deep convolutional neural network for features extraction. In the last stage, they built a three-layer Stacknet of eight selected algorithms using a successive split-grid search for classification and churn prediction. The proposed model obtained the best accuracy score of 83.4%, better than the other proposed models in the literature.

KEYWORDS

Customer Churn, Deep Convolutional Neural Networks, DeepInsight, Halving, Machine Learning, Stacknet

INTRODUCTION

Due to the increased competition in the telecommunication industry, customer churn has become a serious issue for major telecommunication companies. *Customer churn* is the process of customers switching from one service provider to another anonymously (Umayaparvathi & Iyakutti, 2012). Finding a solution for customer churn means that companies should be able to predict in advance customers who are more likely to leave. Retaining customers has become a strategic point to protect revenues for telecom sector investors. To address this situation, telecom operators are forced to take required retention policies and spend more amount of investment on retaining existing customers as opposed to acquiring new ones (Idris & Khan, 2012).

DOI: 10.4018/ijwltt.300342

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Unlike terabyte databases, which typically contain images or multimedia streams, telecommunication databases mainly contain numerous small records describing transactions and network status events (Koutsofios et al., 1999). These facts make the analysis of such massive data a big challenge that consumes a lot of time and needs considerable effort and expertise. The main obstacle to dealing with such a situation is the time necessary to do the treatment and deduct the potential churners; the allowed time should not exceed a few days, to anticipate the departure of the client and give the marketing department time to offer alternatives likely to retain customers. It is important to note that the investment for customer retention is very critical, and errors in detecting real potential churners can cost companies considerable amounts.

Customer churn prediction is mainly a classification problem. The objective then is to predict churners before they leave the company. We consider binary the labels *target*, *Churn*, and *Not Churn* for every customer. It is important to note that the proportions are not equivalent and that a minimum percentage is likely to churn if we consider a random sample of customers. This phenomenon is called *unbalanced datasets*. To process those customers' data and arrive at a realistic model, it is often important to consider a large number of attributes to give visibility on all the factors that can cause a customer to churn.

The approaches and criteria for choosing the best model are very diverse. Hargreaves (2019) employed the Logistic Regression model, as it requires little running time compared to other complicated machine learning algorithms and its output is also easy to interpret. Induja and Eswaramurthy (2016) put into use the kernelized extreme learning machine (KELM) algorithm, proposed to categorize customer churn patterns in the telecom industry. The primary strategy of the proposed work is to organize the data from the telecommunication mobile customer dataset. The data preparation is conducted by using preprocessing with the expectation-maximization (EM) clustering algorithm.

Other than studying a combination of methods and algorithms, Kamalakannan (2018) used a normalized k means algorithm for dataset preprocessing and then selected the attributes from a preprocessed image using the minimum redundancy and maximum relevance (mRMR) approach. It tends to select attributes with a high correlation with the class (output) and allow correlation between themselves. The prediction is examined with the help of a support vector machine with particle swarm optimization (SVM with PSO). Gunay and Ensari (2019) analyzed well-known machine learning methods, including logistic regression, naïve Bayes, support vector machines, and artificial neural networks, and proposed a new prediction method. Abbasimehr et al. (2014) performed a comparative assessment of four major ensemble methods (bagging, boosting, stacking, and voting) based on four base learners, i.e., decision tree, artificial neural network, support vector machine, and reduce incremental pruning, to produce error reduction (RIPPER). The training process was made using the synthetic minority over-sampling technique (SMOTE). The research shows that ensemble learning techniques brought an important improvement to the predictions.

Wangperawong & Brun (2016) represented temporal behavioral data as an image in order to perform churn prediction through deep convolutional networks and used autoencoders for feature importance understanding. Mena et al. (2019) made a comparative analysis on various model architectures to assess the performance of neural networks for churn modeling using recency, frequency, and monetary value. Dingli and Marmara (2017) implemented and compared deep learning algorithms like convolution neural networks and restricted Boltzmann machines to reach very satisfactory results. Elaanba et al. (2021) used deep convolutional neural networks for automatically detecting abnormally positioned tubes and catheters in chest x-ray images.

In summary, the objective of this paper is to develop a churn prediction system capable of quickly detecting potential churners with a high level of certainty and retaining existing customers. The majority of churn classifiers are based on machine learning algorithms and do not exploit the powerful classifier based on a deep convolutional neural network, since the inputs are not images. Our approach distinguishes itself by the use of DeepInsight to transform features into images and

then combine for the first time the strength of convolution neural networks for attribute extraction and Stacknets (Rabbah et al., 2020), which have proven exceptional performance in several prediction problems. This paper gives a brief description of the problem, describes the analytical methods used in this work, presents the methodology, and shows the experimental results, followed by concluding remarks.

PROBLEM DESCRIPTION

The problem of churn management is a priority topic for service providers, especially telecom operators. Accurately predicting future churn is a real challenge for data scientists and business data analysts for several reasons:

- **Number of attributes:** In order to correctly identify this issue, it is necessary to take into account all the factors that can impact end customers. Indeed, it is important to gauge customer satisfaction according to several analysis axes, including the quality of network connectivity, interactions with customer service, consumption and recharge rate, and customer category.
- **Customer behavior:** It is generally not obvious to anticipate unpredictable customer behavior and to take it into account in the modeling of churn prediction mechanisms, since customers often change their behavior unexpectedly, either by changing their offer or for other reasons totally related to the customer's personal choice, such as passing their SIM card to someone else, traveling or for any other reason.
- **Volume of data:** Every day, millions—if not billions—of CDRs (call data records) and EDRs (event data records) are generated by telecom network switches, depending on their size. These records represent a wealth of relevant information that is difficult or impossible to exploit using traditional databases. It is therefore important to have the means and knowledge to store, clean, use and extract useful information from them. It is also necessary to include other data sources related to customer satisfaction and network quality, or even data from social networks that are also important for this use case.
- **Time:** The time dimension is also very important in managing churn in telecom companies. In most cases, results are often obtained late due to a lack of calculation performance, that marketing teams do not have enough time to intervene and retain the most at-risk customers.
- **Money:** The performance of the prediction model in these types of problems is a very critical parameter from a financial perspective. For example, if an unsubscribe prediction algorithm has a high rate of false positives, it results in considerable losses for the company. These losses are both unnecessary responses to retain customers who did not care to unsubscribe and at the same time let real unsubscribers leave the operator.

The construction of efficient and robust models can be achieved through the use of machine learning techniques with a Big Data approach.

ANALYTICAL METHODS

Transforming Non-Image Data to Images Using Deep Insight

It is very difficult to identify small variations in category data in machine learning classification problems. Extracting the relevant details for class identification is then a key to improving model scores. DeepInsight (Sharma et al., 2019) is a method for converting non-image sample data into images in order to exploit the power of convolution networks and the use of GPUs for any type of data. In order to be able to perform feature selection for N collected samples having d attributes each, the principle

of the method is to find a way to organize similar or correlated attributes in neighboring regions of a two-dimensional feature map in order to facilitate learning about their complex relationships and interactions. The main steps are as follows. The first is to transform the raw features into a 2D integration feature space via a nonlinear dimensionality reduction technique, such as *t*-SNE or kernel PCA. Next, the convex hull algorithm is used to find the smallest rectangle containing all features, followed by a rotation is performed to align the feature map frame into a horizontal or vertical shape. Finally, the raw feature values are mapped to the pixel coordinates of the feature map image.

Deep Convolutional Neural Networks for Feature Extraction

Deep convolutional neural networks revolutionized the field of image classification and recognition. They are able to learn basic shapes in the first layers and more complicated forms in deeper layers. This approach was inspired by Hubel and Wiesel in 1962 from stimuli of the visual cortex; their work constituted a major breakthrough in the field of computer vision. The proposed model used the pre-trained VGG16 deep convolutional neural network to extract and reveal x-ray features that are hidden in the original images (see Figure 1). For this purpose, our features extractor used only the convolution layers to extract a single 25,088-length features vector for each input image. Those features are able to identify different levels of image representation. The fully connected VGG16 classification layers are not used by our model.

The model used consists of 20 pre-trained layers. The first layer indicates the input layer, taking an RGB fixed size of $224 \times 224 \times 3$ pixels. The next 16 layers are the combination of convolution+ReLU and max pooling layers. These layers are part of the pre-trained VGG16 Model proposed in Breiman (1996) and trained on the ImageNet dataset (Freund et al., 1999). ImageNet contains around 15 million annotated images from 22,000 different categories, and VGG16 was able to achieve 92.7% accuracy on ImageNet. Therefore we used the VGG16 model as depicted in Figure 1 for feature extraction as a base model. Then we have applied a transfer learning model using a StackNet architecture trained on two X-Ray image datasets (Van der Laan et al., 2007; Michailidis, 2017). The model we built was based on the TensorFlow library backend (Buitinck et al., 2013) to perform tensor operations, TensorFlow is Google's newer generation of AI learning system based on DistBelief (Wangperawong et al., 2016), TensorFlow is a system that analyzes complex data structures into artificial neural networks for analysis and processing. We also used Keras API, a high-level neural network API written in Python.

Searching for Optimal Parameters With Successive Halving

The optimization of parameters for machine learning algorithms can be done in several ways (e.g., grid search and random search). This process is often very tedious and requires a lot of time and computational resources to obtain the optimal combination of model parameters. In this work, we have used a new design that is ten times faster and allows us to obtain satisfactory results. Indeed, version 0.24.0 of scikit-learn has introduced two new experimental implementations of hyper-parameter optimizer classes in the model selection module based on the concept of successive halving: HalvingGridSearchCV and HalvingRandomSearchCV. They use cross-validation to find optimal hyper-parameters. However, instead of independently searching the hyper-parameter set candidates like the classic approach, their successive halving search strategy starts evaluating all the candidates with a small number of resources and iteratively selects the best candidates, using more and more resources. The default resource is the number of samples, but the user can set it to any positive-integer model parameter. Thus, the halving approach has the potential of finding good hyper-parameters in less time.

Ensemble Learning Using Stacknets

In the field of machine learning, we call *ensemble learning* the construction of a meta-model grouping together several machine learning algorithms in order to obtain better performance that exceeds single algorithm models. In other words, the resulting *ensemble* also represents a supervised learning

algorithm, since it can be trained and used thereafter to generate predictions. From a statistical point of view, the hypothesis represented by the ensemble is not necessarily contained in the species of the hypotheses constituting it. In practice, ensemble models generate better results when they contain a significant diversity of base algorithms.

Figure 1 Model summary of the used features Extraction using VGG16 as a base model

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590880
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590880
max_pooling2d_3 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 4096)	102764544
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 2)	8194
Total params: 134,268,738		
Trainable params: 134,268,738		
Non-trainable params: 0		

There are several methods of ensemble learning. *Bagging* (bootstrap aggregation; Breiman, 1996) uses weak learners in parallel on the basis of a given number of bootstrap sets and combines their results in a deterministic way. Boosting (Freund et al., 1999) performs sequential learning of many weak learners and applies weight to training tuples to help the next classifier to achieve better

performance. Wolpert (1992) defined *stacking* as an ensemble learning method, able to combine a set of models built by different algorithms. The first step of stacking is to train different weak learners on the same dataset called the *training set*. The resulting vectors are stacked to a new predictions dataset that represents all first-level predictions. Once the predictions dataset is filled, we train a meta-model to predict the target dataset.

StackNet is a kind of generalization on neural networks to improve accuracy in machine learning (Wolpert, 1992). The activation function can take the form of any supervised machine learning algorithm (e.g., classifiers, regressors, or any ensemble meta-model). The aim of Stacknets is to improve the accuracy of classifiers or reduce training errors. In our case, we used a software implementation of Stacknets in Java. Since it is not always possible to use back-propagation to train Stacknets, as is the case for neural networks, a forward training methodology is essential. In our study, we used k -fold training cross-validation to train our Stacknet.

Let us consider a sample of n independent observations, X_i the input vectors and Y_i the target vectors ($i = 1, 2, \dots, n$). To predict the targets, it is possible to use different types of learners, such as logistic regression, decision trees, support vector machine, and k -nearest neighbor, but it is very difficult to find the optimal learner, and for some cases, learners overfit the training set data.

If we consider M_i^L ($i = 1, 2, \dots, K_L$), K_L number of learners of layer L , and V^L the vector of learners in layer L ($L = 0, 1, \dots, N-1$), N the number of layers. The output of the learner i in the layer $L+1$ could be expressed as follows:

$$\forall 0 \leq L \leq N-1, \forall 1 \leq i \leq K_{L+1}$$

$$O_1^{L+1} = M_1^{L+1}(O_1^L, O_2^L, O_3^L, \dots, O_{K_L}^L) \quad (1)$$

where $P = K_L$ and O_i^{L+1} are the meta-models of layer $L+1$ to combine all previous model's predictions (Michailidis, 2017).

METHODOLOGY

The architecture of our proposed model is shown in Figure 8 and is composed of three stages:

1. feature transformation
2. feature extraction
3. classification and prediction

Stage 1: Feature Transformation

Input Data

The data used for this study is for a telecom subscriber churn dataset consisting of 7,043 customer data with 21 characteristics (3 numeric attributes and 18 categorical attributes), as shown in Table 1. The dataset is public from the Kaggle datasets. The Telco customer churn data contains information about a fictional telco company that provided home phone and Internet services to 7,043 customers in California (BlastChar, 2018).

Data Preprocessing

During this phase, we carried out several transformations of the raw data so that it could be used for the construction of the prediction model. In fact, we first standardized the data by removing spaces and lines with missing information; we also transformed Boolean attributes to contain only zeros

Table 1. Churn dataset features

gender	7043	non-null	object
SeniorCitizen	7043	non-null	int64
Partner	7043	non-null	object
Dependents	7043	non-null	object
tenure	7043	non-null	int64
PhoneService	7043	non-null	object
MultipleLines	7043	non-null	object
InternetService	7043	non-null	object
OnlineSecurity	7043	non-null	object
OnlineBackup	7043	non-null	object
DeviceProtection	7043	non-null	object
TechSupport	7043	non-null	object
StreamingTV	7043	non-null	object
StreamingMovies	7043	non-null	object
Contract	7043	non-null	object
PaperlessBilling	7043	non-null	object
PaymentMethod	7043	non-null	object
MonthlyCharges	7043	non-null	float64
TotalCharges	7043	non-null	object
Churn	7043	non-null	object

and ones and encoded categorical labels. Then, we standardized features by removing the mean and scaling to unit variance.

Feature Engineering

To start our analysis process, we made a selection of features starting from the initial list. This allowed us to reduce the dimensionality and to have the relevant number that will be useful for efficient learning without overfitting. This approach has a significant impact on the simplification of the problem, since the complexity increases exponentially with the dimensionality, and reducing them normally saves processing time and improves the quality of the results.

We performed a mixture of several feature selection approaches, eliminating features with a minimum score for all algorithms. The first method used is a filter-based computation of the absolute value of the Pearson correlation between the target and each attribute. Then, the chi-square metric. The third method used is recursive feature elimination (RFE), consisting of the recursive computation and deletion of attributes at each iteration. We also used built-in methods, such as LASSO, which has its own attribute selection function, forcing the weakest attributes to zero weight, and random forest, which performs selection on feature importance.

In order to maximize the performance of our model, we dropped the least important features. Intuitively, it is possible to explain some cases using the radar graph in Figures 2 and 3. For example, all clients have the feature *Multilines_NoPhoneServ* equal to 0, which means that this feature does not provide any useful information to our prediction model. The same thing happens for *PhoneService*

(equal to 1 for all clients), *DeviceProtection*, *Multilines_Yes* (the same proportion as for *Churner* and *No Churner*).

Table 2. Feature selection results

Features Importance Top 10	Pearson	Chi-2	RFE	Logistics	RandomF	LightGBM
Tenure	1	1	1	1	1	1
TotalCharges	1	1	1	1	1	1
TenureG_Tenure_0-12	1	1	1	1	1	0
MonthlyCharges	1	1	1	0	1	1
InetServ_FiberOptic	1	1	1	1	1	0
Contract_MonthToMonth	1	1	1	1	1	0
Gender	1	1	1	1	0	0
TenureG_Tenure_gt_60	1	1	1	1	0	0
TenureG_Tenure_48_60	1	1	1	1	0	0
TenureG_Tenure_12_24	1	1	1	1	0	0

Imbalanced Class Handling

This type of configuration is often encountered in supervised learning problems, in which there is a large imbalance in the number of observations in the target classes. Churn management is one of those situations in which the number of churners is relatively rare compared to the majority of telecommunications customers. This must be taken into account throughout the data analysis process, from data loading to the validation of the optimal model to be retained. SMOTE (synthetic minority oversampling technique; Chawla et al., 2002) is one of the most famous techniques used to deal with unbalanced data sets. This algorithm oversamples the minority class examples in the training dataset using the k -nearest neighbor algorithm.

Model Building

There is a large choice of families of machine learning models to build a Stacknet, such as boosting (Kim et al., 2018), bagging (Michailidis, n.d.), and neural networks (Michailidis & Soni, n.d.). We built our StackNet model by combining different machine learning algorithms selected from a large set of tested algorithms during a model selection phase.

The optimization of the classifiers was performed through hyper-parameters tuning using the grid search cross-validation approach. To make this choice, a dataset (X_{train} , y_{train}) is used to train the model to be optimized using each combination of hyper-parameters contained in the grid. Then the resulting model is tested on a test dataset (X_{test} , y_{test}), the scores obtained will determine the best optimal combination of parameters of the most efficient model.

Figure 2 Features distribution for Churner

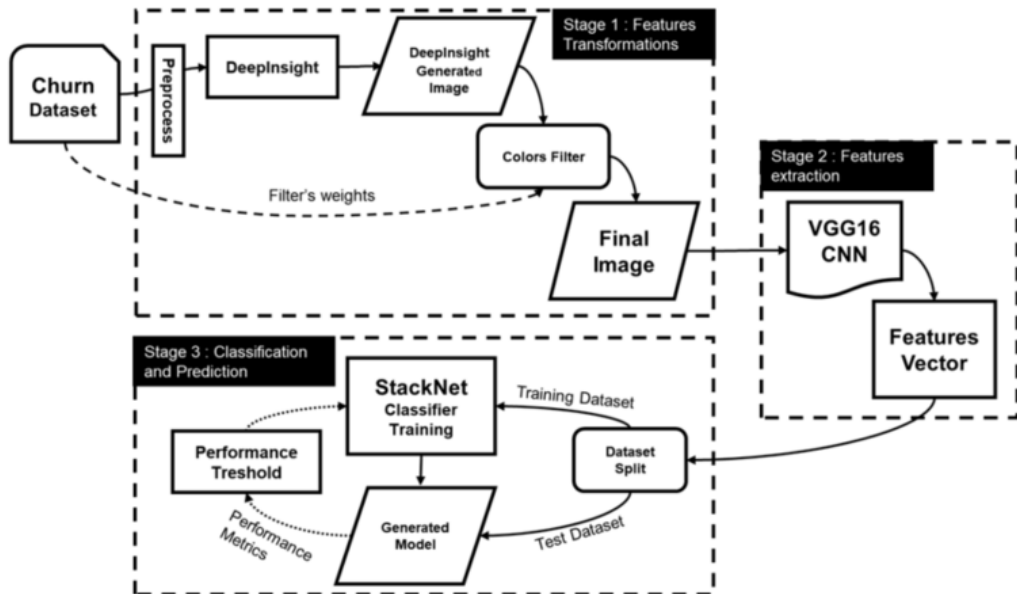


Figure 3. Features distribution for No Churner

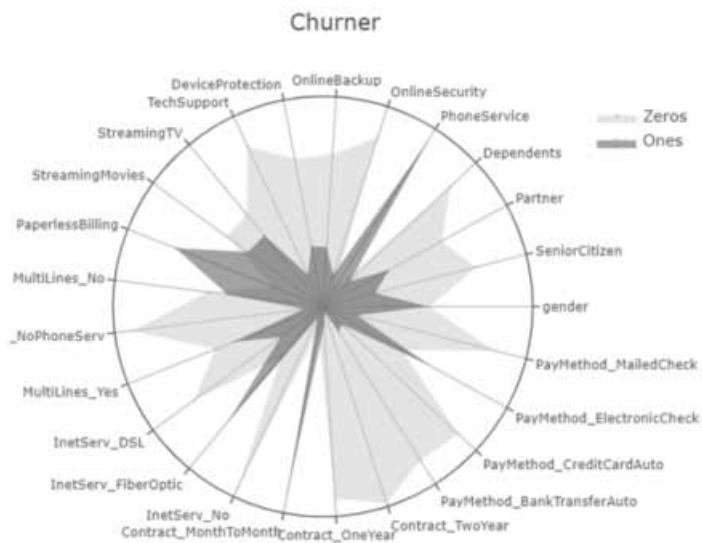
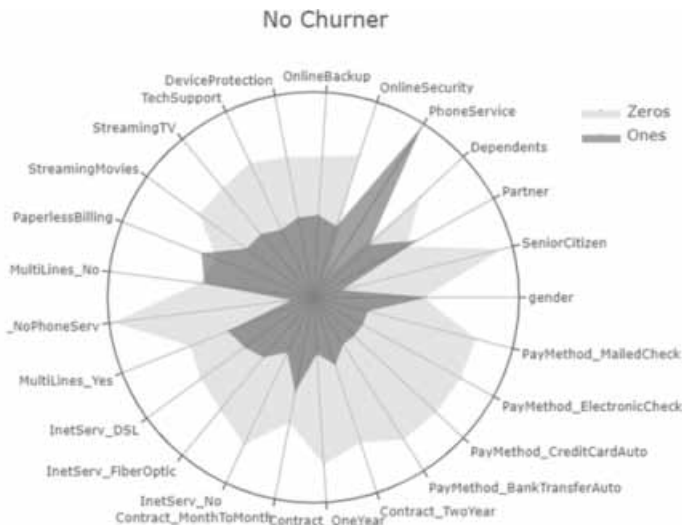


Figure 8 gives an overview of the construction of our churn prediction system. We followed the following steps:

1. **Data cleaning and features selection:** We performed an initial cleanup of the raw data by checking and replacing missing values and correcting any errors in the data type or format. We then checked for outliers.
2. **Polynomial transformation:** To increase the signal or the information included in the set of features, we used a polynomial transformation to generate a more suitable input for the downstream estimators.
3. **SMOT sampling:** As explained above, this technique is used to handle the imbalanced dataset which characterizes our problem.
4. **Image generation:** To identify small data variations, we applied a transformation of the features vectors to images using the DeepInsight method (Sharma et al., 2019). This approach will make

Figure 4 Feature distribution for Churner and No Churner



it possible to exploit the power of convolution networks in the coming phases of our architecture. By entering this step, we have 9,462 samples with 1,044 features each, and the generated image shape is $244 \times 244 \times 3$. Figure 4 shows some samples of generated images.

Stage 2: Feature Extraction

The transformation of our dataset into an image using DeepInsight has allowed for highlighting the correlations that exist between the different features of the initial dataset, as well as revealing the hidden behaviors of the samples. To extract features from the generated images, we used a pre-trained VGG16 convolutional network flattening layer whose role is to apply a flattening transformation on

Figure 5. Models training, validation and testing

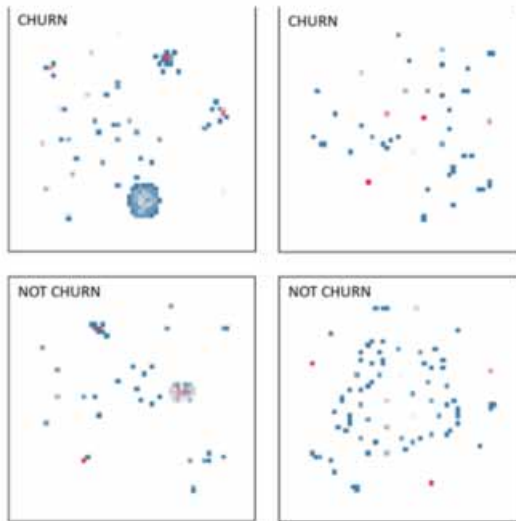
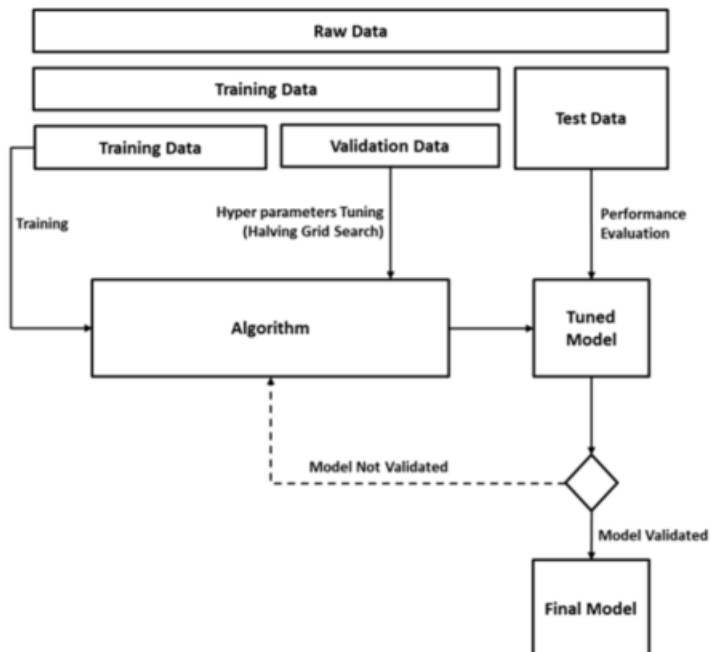


Figure 6. StackNet architecture used to build the churn classifier



the tensor by converting the two-dimensional feature matrix into a vector that can be fed into our StackNet. At this stage, we generated three datasets: training, validation, and testing.

Stage 3: Classification and Prediction

Figure 5 represents our approach to building the StackNet prediction model through an iterative way to obtain the best predictive performance.

StackNet Architecture: In order to build a robust and efficient model, we have opted for the use of a varied set of models based on different algorithms (see Figure 6). The implementation of our solution is done using the Scikit-Learn package of the python language (Pedregosa et al., 2011). We also used an implementation of StackNet called pystacknet, a light python version of StackNet (Michailidis, n.d.; Michailidis & Soni, n.d.), to build our model. The StackNet is composed of five estimators in the first level, three in the second, and one meta-learner in the third level (see Fig. 6).

Evaluation Metrics

To evaluate our approach and estimate the generalization accuracy of our models on new data, we used a set of metrics calculated on the basis of the confusion matrix represented in Figure 7.

Figure 7. Confusion matrix



Figure 8. End-to-end architecture for building the churn prediction model

Predicted	
Actual	True Negative (TN)
	False Positive (FP)
Actual	False Negative (FN)
	True Positive (TP)

Holdout

This approach consists in splitting the dataset into three distinct subgroups, training, validation, and test set (see Figure 5). The training set is used to build the models, the validation set is used in the model performance tuning phase, and the model performance is then evaluated using the remaining test set.

Cross-Validation

This technique consists in dividing the original dataset into k subsamples and evaluating the performance of the model in an iterative way by taking each time one of the k parts of the data as validation and test set, and the remaining $k-1$ parts are used for the model training. This cross-validation technique is called k -fold cross-validation. This technique reduces bias significantly, as

Figure 9 Performance metrics

<div><div>Accuracy</div><div>TP+TN</div><div>TP+TN+FP+FN</div></div>		<div><div>Neg. Precision</div><div>TN</div><div>TN+FN</div></div>		<div><div>Precision</div><div>TP</div><div>TP+FP</div></div>		
		Predicted				
<div><div>Specificity</div><div>TN</div><div>TN+FP</div></div>		Actual	<div><div>True Negative</div><div>(TN)</div></div>		<div><div>False Positive</div><div>(FP)</div></div>	
<div><div>Recall (Sensitivity)</div><div>TP</div><div>TP+FN</div></div>			<div><div>False Negative</div><div>(FN)</div></div>		<div><div>True Positive</div><div>(TP)</div></div>	
			Confusion Matrix			
<div>F1-Score</div>			<div>2 x Precision x Recall</div> <div>Precision + Recall</div>			

Table 3 Performance metrics definitions

Classification accuracy	Is the percentage of correct predictions compared to the set of all predictions
Receiver operating characteristic curve (ROC)	Compare the true-positive rate and the false-positive rate for different classification thresholds for a classifier.
Area undercurve (AUC)	This performance metric consists of calculating the entire two-dimensional areas underneath the entire ROC curve (see Figure 9)
Precision	Gives the level of precision of the model on positive predictions, i.e., the portion of positive predictions that are correct
Recall	Gives the level of precision of the model on the rate true positive predictions compared to the overall number of actual positives
F-measure	Also called F_1 -score, a balance between precision and recall. This measure gives more precise information than the accuracy score, in particular when the problem contains a large number of true negatives, or in the case of imbalanced datasets

most of the data is used for fitting the model. And reduce the variance since all the dataset is used during the test phase.

Model Evaluation Metrics

The choice of evaluation metrics depends on the type of machine learning problem (e.g., classification, clustering, and regression). As we are working on a classification problem, we used the following metrics.

EXPERIMENTAL RESULTS

Test Environment

In order to achieve an optimal prediction result, we have performed the cleaning and preparation of the initial dataset, then we have proceeded to the generation of new features in order to improve the information signal and performed an oversampling to allow more robust learning in our algorithm. After generating the images via DeepInsight, we tested several combinations of basic models for each level. We used a diverse range of models that make different assumptions about the prediction task. The meta-model aims

Figure 10. CovStackNet model architecture

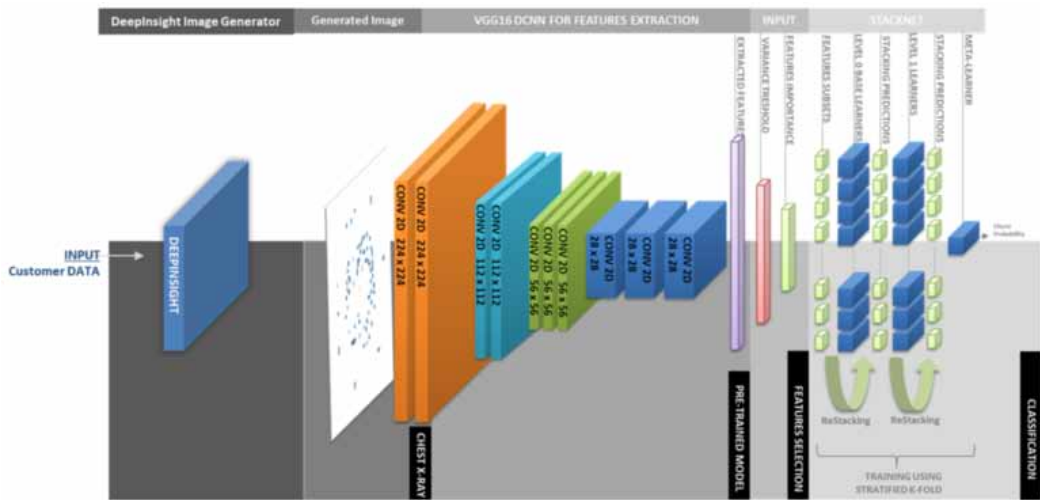


Table 4 Dataset shapes in different steps of the processing pipeline

Steps	Samples	Features
Initial Dataset	7043	21
SMOT Oversampling	9371	21
New Features generation	9371	971
DeepInsight Images	9371	224x224x3
VGG16 features extraction	9371	25088
Features selection	9371	1031

to provide a smooth interpretation of the predictions made by the base models. We then opted for the use of logistic regression as a relatively simple model that still allows us to mix the different predictions.

Figure 10 shows our experimental system design and the sequence of the different steps. All the implementation was done on the Google Cloud Platform (GCP; <https://cloud.google.com>), which allowed us to use the high computational performance of the tensor processing unit (TPU) through Colab TensorFlow Notebooks, the data storage is done through the Google Drive service. Table 4 gives an overview of the size of the data at each processing step. The dataset generated for classification purposes contains 1,031 features.

Figure 11 Classification performance metrics

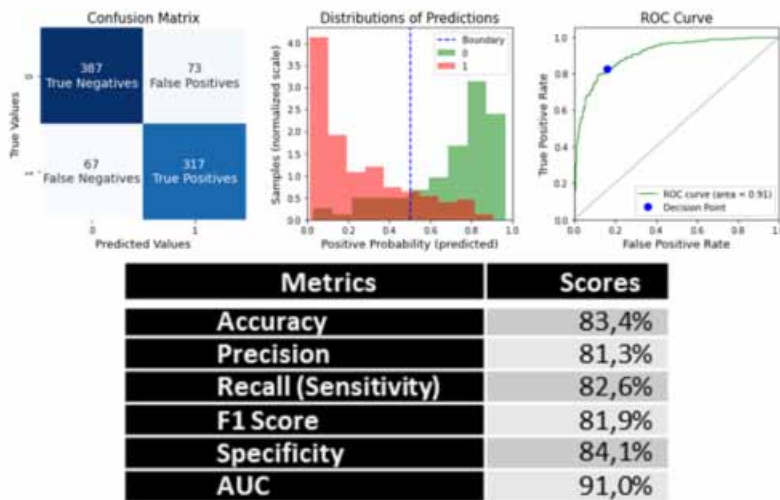
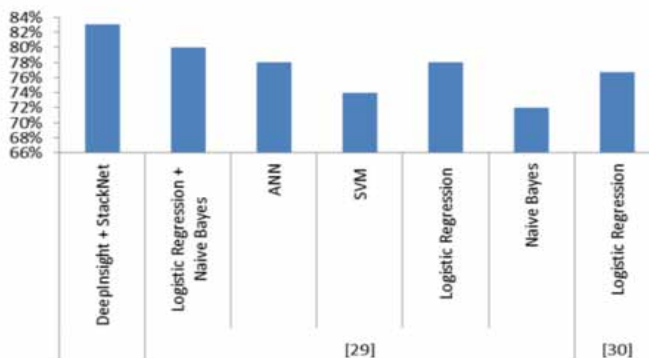


Figure 12 Accuracy scores comparison between our model and other studies results



Experimental Results

From the performance results of Figure 11, we can say that the transformation of attributes into images and the use of the CNN architecture for feature extraction have improved the predictive power of the

classification model based on StackNet. The scores obtained by our approach are more interesting than the results obtained by using classical approaches. Indeed, we can notice, as shown in Figure 12 and Table 5, that the different performances of the models vary from one configuration to another for each approach but remain balanced for our model. As we can see, with a moderate bias and variance, the model becomes more robust and has greater adaptability. In a very schematic way, the model is less sensitive to the training data and with an acceptable bias.

Table 5 Classification metrics comparison DeepInsight + StackNet versus classical approach

Metrics	Out Model	[31	[31	[31	[31	[28
Model		Decision Tree	Logistic Regression	Random Forest	XG Boost	Naive Bayes
Accuracy	83,4%	80%	73%	75%	77%	84%
Precision	81,3%	71%	49%	51%	54%	58%
Recall (Sensitivity)	82,6%	42%	88%	83%	83%	10%
F1 Score	81,9%	53%	63%	63%	66%	18%

CONCLUSION

In this paper, firstly, we have generated images from the preprocessed churn dataset and have used then a pre-trained convolutional neural network to extract a new set of features, and finally, we have used a StackNet architecture to predict the churners. We have compared the different classification performance metrics to understand how those transformations in the data set affects. However, the image transformation has the biggest effect when preprocessing is appropriately done and when the initial dataset is enriched by new features. On the other hand, according to experience, the use of diversified dimensions to build the dataset is strongly recommended. It is possible to include in the same dataset the following information:

- Product information data: service usage, QoS, and coverage
- Marketing data: lifetime customer value, pricing, and customer profiling
- Demographics
- Customer relationship service (CRM) data

In addition, we have proposed a new approach by using transformation of features using DeepInsight to images and tested it with a public dataset. This new method shows the best performance by using a StackNet as a final classifier. As a result, even the new approach has a high prediction rate. It is important to apply it to other datasets with more samples. In general, DeepInsight, VGG16, and StackNet give the highest results, as we expected. Furthermore, from this study, we understand that feature extraction is a key step in our approach. It would be very relevant to test different architectures of pre-trained CNNs, like ResNet, Inception, and Alexnetit.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.

REFERENCES

- Abbasimehr, H., Setak, M., & Tarokh, M. J. (2014). A comparative assessment of the performance of ensemble learning in customer churn prediction. *The International Arab Journal of Information Technology*, 11(6), 599–606.
- Amin, A., Al-Obeidat, F., Shah, B., Adnan, A., Loo, J., & Anwar, S. (2019). Customer churn prediction in telecommunication industry using data certainty. *Journal of Business Research*, 94, 290–301. doi:10.1016/j.jbusres.2018.03.003
- API. (2013). *API design for machine learning software: Experiences from the scikit-learn project* (1309.0238). arXiv.
- BlastChar. (2018). *Telco customer churn* [Data set]. <https://www.kaggle.com/blastchar/telco-customer-churn>
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2), 123–140. doi:10.1007/BF00058655
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- Chuck, U. (2021). *Flatiron phase 3 project: Telco Co. customer churn predictive classification model*. <https://github.com/cutterback/p03-telco-churn-model>
- Dingli, A., Marmara, V., & Fournier, N. S. (2017). Comparison of deep learning algorithms to predict customer churn within a local retail industry. *International Journal of Machine Learning and Computing*, 7(5), 128–132. doi:10.18178/ijmlc.2017.7.5.634
- Elaanba, A., Ridouani, M., & Hassouni, L. (2021). Automatic detection using deep convolutional neural networks for 11 abnormal positioning of tubes and catheters in chest x-ray images. *2021 IEEE World AI IoT Congress (AIIoT)*, 7–12.
- Eleftherios, E. E., North, S. C., & Keim, D. A. (1999). Visualizing large telecommunication data sets. *IEEE Computer Graphics and Applications*, 19(3), 16–19. doi:10.1109/38.761543
- Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Jinkō Chinō Gakkaishi*, 14(5), 771–780.
- Günay, M., & Ensari, T. (2019). New approach for predictive churn analysis in telecom. *WSEAS Transactions on Communications*, 18, 66–70.
- Hargreaves, C. A. (2019). A machine learning algorithm for churn reduction and revenue maximization: An application in the telecommunication industry. *International Journal of Future Computer and Communication*, 8(4), 109–113. doi:10.18178/ijfcc.2019.8.4.550
- Idris, A., & Khan, A. (2012). Customer churn prediction for telecommunication: Employing various features selection techniques and tree based ensemble classifiers. *15th International Multitopic Conference (INMIC)*, 3–27. doi:10.1109/INMIC.2012.6511498
- Induja, S., & Eswaramurthy, V. P. (2016, December). Customers churn prediction and attribute selection. *International Journal of Scientific Research (Ahmedabad, India)*, 5(12), 258–265.
- Kamalakannan, T. (2018). Efficient customer churn prediction model using support vector machine with particle swarm optimization. *International Journal of Pure and Applied Mathematics*, 119(10), 247–254.
- Kim, J., Kim, J., & Kwak, N., (2018). *StackNet: Stacking parameters for continual learning* (1809.02441). arXiv.
- Mena, C. G., De Caigny, A., Coussement, K., De Bock, K. W., & Lessmann, S. (2019). *Churn prediction with sequential data and deep neural networks: A comparative analysis* (1909.11114). arXiv.
- Michailidis, M. (2017). *Investigating machine learning methods in recommender systems* [Doctoral dissertation]. University College London.
- Michailidis, M. (n.d.). *StackNet* [Computer software]. <https://github.com/kaz-Anova/StackNet>
- Michailidis, M., & Soni, A. A. (n.d.). *pystacknet* [Computer software]. <https://github.com/h2oai/pystacknet>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Blondel, M. et al.. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(October), 2825–2830.

Rabbah, J., Ridouani, M., & Hassouni, L. (2020). A new classification model based on StackNet and deep learning for fast detection of COVID-19 through x-ray images. *Fourth International Conference On Intelligent Computing in Data Sciences*, 1–8. doi:10.1109/ICDS50568.2020.9268777

Sharma, A., Vans, E., Shigemizu, D., Boroevich, K. A., & Tsunoda, T. (2019). DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture. *Scientific Reports*, 9(1), 1–7. doi:10.1038/s41598-019-47765-6 PMID:31388036

Umayaparvathi, V., & Iyakutti, K., K. (2012, March). Applications of data mining techniques in telecom churn prediction. *International Journal of Computers and Applications*, 42(20).

Van der Laan, M. J., Polley, E. C., & Hubbard, A. E. (2007). Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1). Advance online publication. doi:10.2202/1544-6115.1309 PMID:17910531

Wangperawong, A., Brun, C., Laudy, O., & Pavasuthipaisit, R. (2016). *Churn analysis using deep convolutional neural networks and autoencoders* (1604.05377). arXiv.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2), 241–259. doi:10.1016/S0893-6080(05)80023-1 PMID:18276425

Jalal Rabbah received his degree of engineer “diplôme d'ingénieur d'état” in telecommunications from National Institute of Post and Telecommunications (Rabat, Morocco) in 2003. He obtained his PhD degree in Applied Mathematics, Computer Sciences and Telecommunication at the Hassan I University (Settat, Morocco) in 2016. He started his professional career as IT engineer at Chauib Doukkali University (El Jadida, Morocco) for two years, and then occupied multiple position at Orange Morocco, first as IT Project Manger at the IT department, Telco Traffic Mediation Expert, and finally as Fraud and Revenue Assurance Expert. And since November 2020 he is Data Science Manger at the Fraud and Revenue Assurance Shared Service (Dakar, Seneg), to manage this risk management activities for Orange Middle East and Africa region. He was certified on ITIL in 2007 and PMP in 2011, ISO27001, Lead Auditor, ISO 21500 Project Manager and PECB Certified Trainer between 2017 and 2019, he is also Six Sigma Green Belt Certified from Orange continuous improvement program. He serves as a reviewer for many international journals. He was a recipient of the 2021 Best Paper Award of the International Conference. His research interests is the machine learning performance optimisation for telco use cases, using Ensembles theory. Also his current research interests are data sciences, especially machine learning applications for healthcare and telecommunication, and big data.

Mohammed Ridouani received the M.Sc in Telecommunications from Faculty of Sciences and Techniques (Fes, Morocco) in 2003. He received his degree of engineer “diplôme d'ingénieur d'état” in telecommunications from National Institute of Post and Telecommunications (Rabat, Morocco) in 2005. He obtained his PhD degree in Applied Mathematics, Computer Sciences and Telecommunication at the Hassan I University (Settat, Morocco) in 2016. He has been an Engineer-Teacher at the Computer Sciences department in High School of Technology at the Hassan II University (Casablanca, Morocco) during 2006-2018, and he was appointed as a professor in 2018. He was certified on CISCO in 2009 and Linux LPIC-1, LPIC-2 and LPIC-3 (senior/expert level) in 2008, 2012 and 2017 respectively. He is a LPI and Cisco instructor and he has served as an international instructor at the Francophone University Agency. He serves as a reviewer for prestigious international journals, and TPC member roles in many international conferences (IEEE GLOBECOM, IEEE PIMRC, IEEE VTC2018-Spring, IEEE ISWCS, IEEE IWCMC, IEEE EuCNC & 6G Summit - 6ET, IEEE 5G-WF, etc.). He was a recipient of the 2021 Best Paper Award of the International Conference on Smart Systems and Data science 2021 (ICSSD'21). His research interests are in the physical and the MAC layers design based wireless communications systems and mathematical modeling, including cognitive radio, resource allocation strategies, compressed sensing, cooperative. Also his current research interests are data sciences and smart city, especially machine learning applications for healthcare and Telecommunication, and Big data.

Larbi Hassouni got his engineer degree in 1983 from the “École Centrale de Marseille”. He prepared his Phd degree in 1987 at the University of Aix Marseille III in France. Among his research work, there is the development of a list unification software with LeLisp language of INRIA in order to contribute to the realization of the inference engine of an expert system for the digital circuits' diagnosis. He also developed a behavioral symbolic simulator of digital circuits using the C, FRL, and LeLisp languages in order to contribute to the development of a “formal proof” tool for correcting a design of material produced by a Hardware Design Language (HDL). Currently, his research work mainly concerns Data Sciences.