Name: NISHANT TIWARI

Email address: nissanttiwari@gmail.com

Contact number: 8602856280

Anydesk address: 366 543 569

Years of Work Experience: **0**

Date: **16th July 2021**

Self Case Study -1: ***Santander Customer Satisfaction***

Overview

1. Introduction:

Santander Bank is a retail banking company, meant for providing banking services to the general public. Customer satisfaction is of utmost priority for any business and it is very important for a bank to identify all the unsatisfied customers so that management can take care of their needs and prevent them from leaving the bank. But the problem lies in identification of these unsatisfied customers because the unsatisfied customers don't express their reason of dissatisfaction with the bank directly. In addition to that it is just not possible for banking management to identify unsatisfied customers by just looking at their banking records because it is very difficult for a human being to identify a pattern from so many variables present in the customer's banking record. Also doing that is not possible for each customer as the number of customers is very high.

So for the given problem employing a machine learning model is the only solution by which unsatisfied customers can be identified in very less time. Using

Machine Learning, a model can be created that can identify unsatisfied customers. After which the bank can directly approach those customers and resolve their issue or take appropriate action like offering them exclusive offers to prevent the customer from leaving the bank.

2. ML formulation:

It is a binary classification problem with the target/class denoting Unsatisfied Customers with '1' and Satisfied Customers with '0'. We need to develop a model which can classify given data points correctly as 1 or 0 i.e satisfied or unsatisfied customer.

Features don't denote anything specific as the column names have been changed, for privacy reasons I guess.

3. Business constraints:

There aren't any business constraints specified anywhere in the problem.

4. Data set analysis:

Data can be obtained from the link:

https://www.kaggle.com/c/santander-customer-satisfaction/data

For the dataset we are given, each row represents a customer. We are given two datasets 'train.csv' and 'test.csv' with 371 and 370 features respectively. Since it is a competition only train dataset has an extra column called 'TARGET' which isn't present in test dataset. 'TARGET' column show's customer's satisfaction. Value '0' in TARGET column means the customer is satisfied, value '1' means customer is unsatisfied. Column names don't really convey a meaning about what they represent, maybe the column names are renamed on purpose for security/privacy reasons.

Data is highly unbalanced with only `3.957%` belonging to class '1' that is unsatisfied customers.

5. Performance metric:

Optimum performance metric for this problem is f1 score as the data is highly imbalanced and choosing ROC-AUC isn't good for highly imbalanced data. So we will use f1-score instead which takes in consideration both recall as well as precision.

Research-Papers/Solutions/Architectures/Kernels

1. https://towardsdatascience.com/santander-customer-satisfaction-a-self-case-study-using-python-5776d3f8b060 Ashish in his solution has done following as

   Exploratory data analysis:

   - Removed features which are constants.
   - Removed duplicate features (two or more features with same values.)
   - Remove sparse features, i.e features which has maximum(90%+) values as 0.
   - Identified some features as mortgage, age, region. Out of which identified that in age feature, customers below 23 were mostly satisfied, so made a new feature which denoted whether the customer's age is less than 23 or not.
   - Identified those features which had keywords like 'imp', 'ind', 'num', 'saldo' and applied log transformation on imp and saldo feature, and created a new dataset.

   Feature engineering:

   - Made two new features that had a count of the number of zeros and non zeroes in all features.
   - Added a feature to calculate number of zeroes and non zeroes across the feature with keywords salado, num, imp, ind.
   - Calculated number of zeroes and non zeroes across keywords 'imp', 'saldo', 'num', 'ind' and added af feature.
   - Took the average value of those 'saldo' and 'imp' features which had unique values between 50 and 210 and added it as a feature.

- Used ‘K_means cluster’ with 5 values of K and after getting predictions corresponding to 5 values of K used them as features.
- Removed features with correlation greater than 0.95 with each other and removed feature having correlation less than 10**-3 with TARGET.
- Created separate dataset with log transformations to feature ‘var38’, ‘saldo’, ‘imp’.
- Applied One hot Encoding for the features that had values between 3 to 10.
- Standardized all datasets, used PCA and added as feature

Models:

- Used random forest and selected top 250 features for each dataset.
- Used Random Search CV on Logistic Regression, Decision Trees and Random Forest models to find optimum hyperparameters.
- Manually selected hyperparameter for XGBOOST and LightGBM.
- Then picked best models from different datasets for ensembling and stacking.
- Ensembling of the two models- ‘XGB model fitted on Log transformed Response Encoded (top 250 features) dataset’ and ‘XGB model fitted on Normal response encoded (top 250 features)’ performed the best.

IMP POINTS: Age (var_15_0) turned out to be of high feature importance that means this feature can be further used in feature engineering.

2. http://gh.mltrainings.ru/presentations/Efimov_SantanderCustomerSatisfaction.pdf
Lucas and team has followed following procedure:
    - Removed columns which were constants(only zeroes)
    - Removed duplicate columns
    - Calculated Sum of Zeros
    - Used K-means and added number of clusters as feature
    - A binary feature denoting whether account is active or inactive
    - Models: Trained different models such as XGBoost, RGF, Neural network, FTRL, Random Forest, Adaboost, ExtraTrees, KNN, Lasso, SVM on different set of train data to reduce noise (Bagging technique).
    - Final prediction was made as a linear combination of predictions from the above models.

3. https://www.kaggle.com/solegalli/feature-selection-with-feature-engine Here in this paper Sole has shown how the features can be processed in a very simple manner using a library called feature_engine.selection. This library is capable of removing duplicated features, constant features, correlated features.

   Using SmartCorrelatedSelection one can select important features based on missing values, variance of the features, importance of feature as per a model defined, cardinality.

   A pipeline can also be defined for performing all the above activities and select features.

   SelectByShuffling Initially takes in all the features and starts dropping feature one by pne. If there is considerable drop in performance of model that means the feature is important

   SelectBySingleFeaturePerformance makes one model for a feature and those models are selected which gives model's performance more than a defined threshold

   RecursiveFeatureElimination First takes in all features and trains a model. Then feature importance for that model is obtained, least important features are removed. New model is trained and performance is obtained. If the drop of performance is very high then the feature is kept otherwise it is removed.

   IMP POINTS: feature_engine.selection is a fast way for feature selection and might come in handy for this dataset. However, solely depending on this library for feature engineering won't be a good idea because the AUC obtained by this kaggler isn't very high as compared to others for this competition.

4. https://www.kaggle.com/adarshsng/extensive-advance-feature-selection-tutorial

   Incorporated two of the feature selection process 'ExhaustiveFeatureSelector'

and ‘SequentialFeatureSelector’ from ‘mlxtend.feature_selection’ for brute force evaluation of features.

Recursive Feature selection using RFE by sklearn.feature_selection. In this estimator is trained on set of feature and importance of feature is obtained and least important features are removed. This process is repeated until we have desired number of features remaining.

IMP POINTS: Using two feature selection processes ‘ExhaustiveFeatureSelector’ and ‘SequentialFeatureSelector’ by mlxtend.feature_selection seems a lot efficient as it computes performance based on all combination features can have together. However it may seems time consuming so these should be done only when features have already been filtered out manually and have less number of features.

5. https://beta.vu.nl/nl/Images/werkstuk-elsen_tcm235-865964.pdf Feature names are anonymous so author of this paper suspects some features to convey meaning such as ‘var3’ as nationality, ‘var_15’ as age, ‘var_38’ as mortgage value, ‘Num_var4’ as number of bank products.

   In addition to removing duplicate features, duplicate observations are also removed.

   IMP POINTS: Great deduction of anonymous features to meaningful feature which might turn out to be useful in feature engineering.

6. https://www.kaggle.com/cerolacia/customer-satisfaction Used oversampling using SMOTE for balancing the dataset.

Used PCA for Dimension reduction. But her auc score isn't as good as others, maybe because she didn't do any feature selection manually and a lot of unnecessary features ruined the score despite hyperparameter tuning.
IMP POINTS: Can't completely rely on libraries for feature selection, manually selecting features will be beneficial for this dataset.

First Cut Approach

1. **PREPROCESSING AND FEATURE SELECTION:**
    - Most of the papers and solutions mainly focus more on feature engineering and less on modelling, so for this dataset features processing and engineering is of utmost importance which influences the score of the model.
    - Although I encountered only one solution which used oversampling technique for balancing the class, I would experiment with Oversampling and undersampling techniques by using 'RandomOverSampler', 'RandomUnderSampler' and a combination of both and see how the result turns out with and without oversampling/undersampling.
    - Going through existing solutions, solely depending on dimension reduction such as PCA, didn't end in a good AUC score.

      Initially, manually unimportant features must be removed, then we can use other feature selection methods that too in moderation. Solutions for this problem, which only used feature selection aggressively resulted in poor AUC at the end.

    - Based on most of the solutions above, it is evident that the dataset has duplicates, constant features, sparse features so initial approach would be to remove duplicate features and also remove those features which are constant. Furthermore Quassi Constant features that don't show much variance can be removed as well. Furthermore we also need to check for duplicate observations and eliminate them as well.

- As the number of feature is reduced we can further filter important features by using ‘ExhaustiveFeatureSelector’ and ‘SequentialFeatureSelector’.
- New features can be added such as count of the number of zeros in all features for each row.
- Feature var_15 has been identified as age in some papers and also an important feature in classification. It is also evident from some papers that people below age 23 are always satisfied, so this feature can be transformed as a categorical feature based on whether age is more or less than 23. Similarly we can also check if there is any age above which customers generally tend to be unsatisfied. If there is such an age we can make another categorical feature based on it.

2. **MODELING:**
    - There isn’t any one model which will give the best result, ensembling technique has given the best result out of all the solutions. So it will be best to use ensembling with various models as well.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument “X” (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):
        preprocess data i.e data cleaning, filling missing values etc
        compute features based on this X
        use pre trained model
        return predicted outputs
    final([time, location])
    d. in the instructions, we have mentioned two functions one with original values and one without it

    e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)
4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models