Data-Intensive Project: Predicting Customer Churn

Kailin Wu and Christian Durán García
KTH Royal Institute of Technology
October 21st 2023

The Project

We have produced a Machine Learning model that is capable of predicting whether a client is going to churn or not. We made all the steps to carry on a data analysis project in a single Jupyter Notebook using Spark, SparkSQL, and SparkML; also other additional tools available in python like Matplotlib, Seaborn, and Pandas for compatibility for the previous two libraries.

Our project consisted in the traditional data science methodology of Data Collection, Exploratory Data Analysis, Feature Engineering, Model Training and Selection, and finally Evaluation. For each of this sections we performed the following tasks:

- Data Collection: We searched on popular data science websites such as kaggle for datasets that fit our expectations for the project, the dataset we found consisted in data available right away for classification efforts. This dataset is split into training and evaluation, so that we can test our produced models with unseen data.
- EDA: We did some analysis in order to explore and comprehend the behavior of each feature in both datasets, mainly in regards to the Churn feature, which is our label feature for classification. With the interpretation of such analysis we managed to gather the necessary information to make a decision on what kind of feature engineering to make in order to have models that are able to generalize to the evaluation dataset.
- Feature Engineering: We performed some feature engineering in all the variables applying the next techniques: Feature transformation for some numerical features, we did a logarithmic transformation so that we can overpass some skewness in the data. Binning in some other numerical features to achieve similar results as with the other technique. Interaction Features by encoding categorical features and then applying a simple multiplication between them to create some interaction and capture those.
- Model Training and Selection: We choose to try four different models: LogisticRegression, RandomForestClassifier, NaiveBayes, and LinearSVC for our classification experiments. We fed the engineered features to these models and applied a hyperparameter-cross validation approach for training in order to find the most optimal version of each model, scoring them with custom made functions for binary classification reports and confusion matrices. Then we tested the model with a dataset we excluded from the original training dataset and got pretty optimistic results, with accuracy numbers ranging from 0.86 to 1.00.
- Evaluation: In order to check if these models are able to generalize to unseen data we evaluated them with the evaluation dataset and got some interesting results. Both LinearRegression and RandomForestClassifier got decent accuracy results of 0.89 and

0.86 respectively, but weren't able to predict the label 0 class. NaiveBayes did the worst with 0.69 accuracy and a recall for label 0 class of 0.41. LinearSVC on the other hand managed to get perfect results, meaning that the model was able to generalize well to unseen data and that the classes are linearly separable with the correct feature engineering.

Method

Our method is quite simple and it is based on the traditional data analysis pipeline for a project. It consisted in five parts, each one with its unique tasks at hand.

- Data Collection: Getting a dataset with enough records so that it can be justifiable using a data intensive platform like spark, our dataset has half-million records.
- EDA: By doing some descriptive statistics, correlation analysis, and some visualizations with countplots, kde plots, and histograms, we want to have a broad perspective on each feature to future decision making for feature engineering.
- Feature Engineering: Apply some transformations on the raw data in order to better capture the behavior and be able to produce models with generalization capabilities.
- Model Training: Train different models in a cross validation way with hyperparameter tuning to get the best model for the training and testing data.
- Evaluation: Generate predictions with the evaluation dataset to monitor whether the models are able to generalize to unseen data.

Results

We managed to produce a LinearSVC model that generalizes well to unseen data, we achieved this through feature engineering numerical and categorical into transformation, binning, and interaction features that better capture the behavior of the data, both in the training and evaluation dataset. This model surpasses the performance of the other tested models, indicating that the features, engineered the way we did, are linearly separable and suitable for a classification model.

After generating the predictions, we created other features for the evaluation dataset with the respective model's predictions in order to visualize them through a BI tool like PowerBI, so that results are easily presentable to others.

Conclusions

By using the Spark framework and some of its most important tools like SparkSQL and SparkML we now have added some new features to our data science stack. Our main takeaway from this project is the knowledge of how to use these tools in python and why they are important for large scale data processing. In our experience using pandas and other packages like numpy, we have encountered some concerns when dealing with huge datasets, something that Spark takes care of. The only thing we point out when using this framework in a single computer is that collecting and visualizing data is certainly not the purpose for

Spark, because collecting data is a costly operation when working with distributed data, but when applying transformations to the data and training models on huge datasets Spark is king.

How to run the code

Our complete project is implemented in a single Jupyter notebook, in order to run it there are some requirements for it.

- Install anaconda.
- Install openjdk, pyspark, findpark (and the other packages in the case they are not installed just yet). This way we can run spark applications directly using the anaconda environment in any os (we tested in Windows and Mac M1).
- Hit Run All in the notebook and it should start executing each cell at a time. There are some markdown cells as well that link the cells in order to make a cohesive story along the way.
- The resulting evaluation dataset is going to be stored in a csv file in the folder where the notebook is running.

The code is available in the next github repository:
https://github.com/Christian-Duran-Garcia/Data-Intensive_Project