Proposal

Machine Learning Engineer Nanodegree

Capstone Proposal

Babs Khalidson, 5th October, 2022

The project is a Kaggle competition, Customer Churn Prediction 2020

Domain Background

This competition is about predicting whether a customer will change telecommunications provider, something known as `churning`.

Problem Statement

Customer churn is a common problem for businesses that provide subscription services. It is often difficult to determine when exactly a customer is likely to churn, due to the fact that it could be caused by several reasons.

The challenge is to predict whether a customer will churn or not using 19 input features defined in the dataset. The accuracy of the model will be the main metric for determining its success.

Datasets and Inputs

The training dataset contains 4250 samples. Each sample contains 19 features and 1 boolean variable `churn` which indicates the class of the sample. The 19 input features and 1 target variable are:

File Descriptions

All of these files can be found in `dataset` folder within the submission:

- **train.csv** - the training set. Contains 4250 lines with 20 columns. 3652 samples (85.93%) belong to class churn=no and 598 samples (14.07%) belong to class churn=*yes*
- **test.csv** - the test set. Contains 750 lines with 20 columns: the index of each sample and the 19 features (missing the target variable `churn`).
- **sampleSubmission.csv** - a sample submission file in the correct format

Data Fields

1. `state`, *string*. 2-letter code of the US state of customer residence
2. `account_length`, *numerical*. Number of months the customer has been with the current telco provider
3. `area_code`, *string*=`area_code_AAA` where AAA = 3 digit area code.
4. `international_plan`, *string*,(yes/no). The customer has international plan.

5. `voice_mail_plan`, *string*, (yes/no). The customer has voice mail plan.
6. `number_vmail_messages`, *numerical*. Number of voice-mail messages.
7. `total_day_minutes`, *numerical*. Total minutes of day calls.
8. `total_day_calls`, *numerical*. Total minutes of day calls.
9. `total_day_charge`, *numerical*. Total charge of day calls.
10. `total_eve_minutes`, *numerical*. Total minutes of evening calls.
11. `total_eve_calls`, *numerical*. Total number of evening calls.
12. `total_eve_charge`, *numerical*•. Total charge of evening calls.
13. `total_night_minutes`, *numerical*. Total minutes of night calls.
14. `total_night_calls`, *numerical*. Total number of night calls.
15. `total_night_charge`, *numerical*. Total charge of night calls.
16. `total_intl_minutes`, *numerical*. Total minutes of international calls.
17. `total_intl_calls`, *numerical*. Total number of international calls.
18. `total_intl_charge`, *numerical*. Total charge of international calls
19. `number_customer_service_calls`, *numerical*. Number of calls to customer service
20. `churn`, *categorical*, (yes/no). Customer churn - target variable.

Solution Statement

The problem is a classification problem which will require using machine learning algorithms to predict the classes.

In this project, I propose using AutoGluon for this supervised learning task on tabular data. With AutoGluon I plan on automating the data cleaning, feature engineering, model selection and hyperparameter tuning to find the top-performing model that would be ready for production.

Benchmark Model

The benchmark model that I will compare my solution to would be the `BaggingClassifier` produced by another kaggler that achieved an accuracy of 96% accuracy on this dataset.

Evaluation Metrics

The evaluation metric for this competition is the test accuracy, defined as follows:

$$\text{Accuracy} = \frac{\text{Number of correctly predicted test samples}}{\text{Total number of test samples}}$$

Project Design

The workflow for approaching a solution:

1. `Data Analysis`: understand the datasets
2. `Features Transformation`: convert variables into features. Standardize/normalize features, apply numerical transformations
3. `Features Selection`: select relevant features
4. `Machine Learning Models`: train different models using AutoGluon. Perform hyperparameter

5. `Evaluation`: evaluate the performance of each strategy, and check possibilities of combining them to extract the best of each one and achieving an optimal model.
6. `Deployment`: deploy the trained model to an AWS endpoint
7. `Lambda & Step Functions`: set up a AWS Lambda & Step Function for calling the deployed model.
8. `Web App`: Create a web app by deploying the model on streamlit or flask.