Customer Revenue Prediction

Domain Background

Any company wants to increase the revenue by minimizing the investment. Only a small proportion of the customers produce most of the revenue. Knowing this proportion is essential for marketing teams to make appropriate investments in the promotional strategies. The forecast can be based on past sales data, industry wide comparisons and economic trends. Predicting revenue not only enable how to manage the workforce, cash flow and resources but also helps the company when looking to acquire investment capital. Revenue forecast allows you to spot potential issues while there's still time to avoid or migrate them. According to research from the Aberdeen Group, companies are more likely to grow revenue by 10% year-over-year when they have an accurate revenue forecast system.

Problem Statement

The main objective of the project is to apply Machine Learning techniques to accurately predict revenue per customer.

Given a historic sales and associated revenue information, we want to be able to predict the revenue of a customer in future time period. The accuracy is determined by the Root Mean Squared Error(RSME). Lower the value more accurate is the prediction.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where $\hat{y}$ is the natural log of the predicted revenue and y is the natural log of the actual summed revenue for a customer.

Datasets and Inputs

The dataset is provided by RStudio on Kaggle competition on Kaggle includes approximately 2 million visits to the store separated into training and test datasets. The data is split training and testing sets. The training set contains 1.7 million visits to the store, each has the

channelGrouping, customDimensions, date, device, fullVisitorId, geoNetwork, hits, socialEngagementType, totals, trafficSource, visitId, visitNumber and visitStartTime. I will be splitting the training set into 80% training and 20% testing datasets. I will separate the revenue generated from the testing dataset so that we can compare them with the predicted values.

Solution Statement

The solutions is a time series classification model capable of capturing patterns in the training data and therefore can be used to make predictions regarding the future trend of data. First I will be using pandas and numpy to gain some understanding of the data, then try to device some features based on the given features to train and remove features that does not help in training. As for the model, I am inclined towards using Long Short Term Memory Networks(LSTM), an artificial recurrent Neural Network, that has proven capabilities in classifying, processing and making predictions based on the time series data.

Benchmark Model

I will be using statsmodel.tsa's AR Model as my benchmark model for predicting the revenue and compare those predictions with the predictions of the model I created. I will be comparing the accuracy score of the models to show which model is better performing. I will run both the models with the same dataset(train/test).

Evaluation Metrics

The model is evaluated based on the RSME. Since the test data is not labelled, evaluation is done by using the training dataset by separating the revenue into separate file and comparing the results using RSME.

Project Design

- **Programming Language**: Python 3.6
- **Library**: Pandas, Numpy, Pytorch, LSTM
- **Services**: Jupyter Notebooks, Sagemaker
- **Workflow**:
    - **Data loading and exploration**: Visual representation of the dataset and preparing the data in the required format for the model.
    - **Data preprocessing**: Scaling and normalization operations on data and splitting the data into training and test datasets.

- **Future Engineering**: Finding relevant features, engineer new features using methods like PCA if needed.
- **Model Training**:Train the basic classification model on the training set and gauge the performance
- **Evaluating Model**: Evaluating the model against the test dataset using RSME to determine the accuracy of the predictions..
- **Model Tuning**: Fine tuning the hyperparameters to increase the model performance, if necessary.