AIDI 2005 – CAPSTONE TERM 2

MARCOS BITTENCOURT

Modelling Part I

AHNCH BALA 100424062

SONAKSHI KARKERA 100720763

SURBHI THAKUR 100732335

ARUN KALAESWARAN 100771700

FEBRUARY 21, 2020

Model Structure: Evaluating Models

Structure of our model will consist of different layers like data loading, data preprocessing, testing and training and output. The training and testing models are described below.

Multiple Linear Regression:

Looking at the problem at hand, along with the data that has been provided to us, we can easily conclude that the simplest path to tackle the issue is to use a multiple linear regression model. In our problem we have various independent variables and one dependent variable. A linear regression model can fit the best linear relationship between the independent and dependent variables. Although we feel this might not be the most accurate model that can be used, it will give us a better understanding of the issue at hand without implementing complex code.

LightGBM (LGBM):

LightGBM is a gradient boosting model that is based on algorithms that use tree learning. The difference between LightGBM and other tree-based models is that LightGBM grows trees vertically, while other algorithms grow trees horizontally. The leaf with the maximum delta loss will be the choice of growth for LGBM models. This helps reduce more loss compared to horizontal tree growing models.

LGBM models are a great choice for regression models like ours especially when there is a large dataset. LGBM can handle large sizes of data with ease and can process it extremely fast. Another reason why this framework is great is because it focuses on maximizing the accuracy of its results. LGBM, however, does tend to overfit on small datasets.

Since we have an abundance of data and our primary focus is being as accurate as possible, LGBM fits the bill. The implementation is easy and is not as computationally expensive as other models on a dataset of our size.

Some key control parameters to look at:

- max_depth: Describes the maximum depth of the tree. This will help deal with overfitting, if there is any (lower to combat overfitting).
- min_data_in_leaf: Defines the number of records a single leaf can have.
- lambda: This parameter specifies regularization.
- early_stopping_round: This parameter optimizes the speed of our analysis. If one of the metrics does not improve from the last early_stopping_round, then the model will stop training.

Core Parameters we can look at:

- Task: specified whether we should train or predict on our data
- Application: specified whether our problem is regression or classification. In our case it is regression, which is the default selection for Light BGM.
- Boosting: defines the type of boosting that will be applied by our framework. The boosting offered are Gradient Boosting Decision Tree, Random Forest, Dropouts meet Multiple Additive Regression Trees and Gradient-based One-Side Sampling. We believe Gradient Boosting Decision Tree will be enough for our problem.
- num_boost_round: Specifies how many iterations of boosting we will need.
- learning_rate: This is important as it decides how much of an impact each tree will have on the final outcome.
- num_leaves: Number of leaves in a full tree. The default is 31.
- device: default is set to CPU, but we can attach GPU's as well.

Neural Network:

Keras is a high-level neural network API written in Python. We use the pandas package to manipulate data, and we have implemented Tensorflow's Keras (Google's machine learning software) to create a neural network. The general structure of ANNs contains neurons, weights, and bias. Based on their powerful molding ability, ANNs are still very popular in the machine learning field. However, there are many ANN structures used in the machine learning problems.It allows easy and fast prototyping. It runs effectively on GPU and CPU. It can easily support both convolutional network and recurrent networks.

Below is the demonstration of Neural Networks.

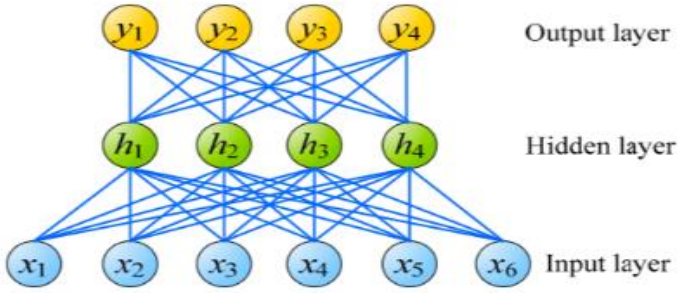
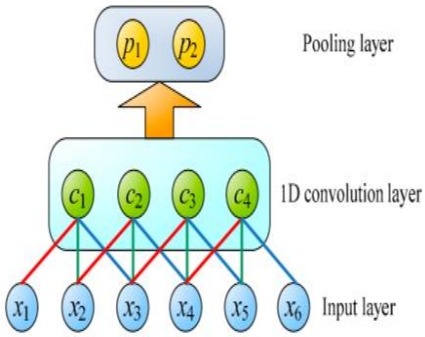


Figure 1. The Multilayer Perceptron (MLP) structure.



Software Tools

- For this project we will be using jupyter notebook as our IDE.
- For the database we will use Microsoft Office Suites for data storage.
- We will be using Dash which is python framework for building our analyitical web application.
- Python Libraries - Keras, Tensorflow, Numpy, Pandas, Matplotlib, Bokeh

Calibration of Model Techniques

After applying our machine learning algorithms and testing the accuracy, we will try improving the results by: -

- Feature Scaling: by normalizing the independent variables , we help to speed up the calculations later when ML algorithms are applied.

- Class Imbalance: By reducing the class imbalance that we see between the electric meter readings and the other readings; we will notice less skewness for the other meter readings.

- Cross Validation / Hyperparameter Tuning: By changing certain features of the ML code and reducing the error in our training set will yield us better results in our testing data.