**Contest fORged by Amazon Web Services and cORe**

**Amir Mohammad Esmaieeli Sikaroudi, PhD student of Systems and Industrial Engineering, University of Arizona, Tucson, amir.esmaieeli@email.arizona.edu**

**Md Habibor Rahman, PhD student of Systems and Industrial Engineering, University of Arizona, Tucson, habiborrahman@email.arizona.edu**

**Introduction**

Our goal is to develop an inventory model for a hardware device, by leveraging past demand data of that product. Despite it’s being called an inventory problem, after all, the main task is demand forecasting. Compared to a traditional inventory problem, there is no ordering cost, no capacity constraints for inventory, and no consideration of reorder point, although there is delivery lead time of one month, i.e., we will receive the ordered quantity at the beginning of next month if we place that order at the starting of current month. However, if we can predict the demand for the next month, we can easily calculate how much we need to order. Then we can fulfill the sales for the upcoming month with the received quantity. In a nutshell, the cost will be minimum when the forecasting accuracy is maximum.

In this data driven approach, we are utilizing past monthly demand data for forecasting the demand of future months, with the step of one month. We will utilize the information of n months to predict the demand on (n+1)th month. Moreover, we will be updating our data on a monthly basis, once real demands are being available during future months.

**Insights gained from the data provided**

In our data driven modelling approach, previous demand and cost data played a major role. We gained several key insights from the provided dataset. They are as follows:

1. Over years, the demand is gradually increasing, i.e., we can observe an increasing trend. However, starting from the last quarter of 1999, specially in the year 2000, the demand boomed. Probably, there were some assignable causes behind this. For instance, some new technology might have been introduced that immediately increased the demand for that hardware device. After 2000, the demand dropped to its initial pattern.
2. There is another monthly demand pattern which is similar across the years. For instance, during March, June, September, and December (3rd, 6th, 9th, and 12th month in a year) the demand increases. Additionally, there is a sharp drop in August (8th month) every year.
3. We also observed that it is always preferable to incur holding cost instead of backorder cost. While holding cost is $1 or $2 at max, backorder cost is $3 per product per month.

**Review of literature (we can change this name if we don’t mention about any literature)**

ARMA: Auto regressive moving average (ARMA) can describe stochastic processes in the form of two polynomials, one polynomial for autoregression and one for moving average. ARMA model is a very helpful tool for prediction (forecasting) of future values while analyzing time series data.

ARIMA: A generalization of ARMA model is autoregressive integrated moving average (ARIMA), which comes handy if the data has non-stationary properties. In ARIMA, the AR part refers to regressing the variable of interest on its prior values, Integrated part helps to eliminate non-stationarity in data, and MA shows the regression error as a linear combination or past errors.

Multilayer perceptron: MLP belongs to a group of feedforward artificial neural network that leverages backpropagation, a supervised learning technique, for training. MLPs can solve problems stochastically and can be utilized to create classifier algorithms as well as mathematical models by regression analysis.

Random forest: Random forests refer to an ensemble learning scheme used for classification and regression. Random forests also rectifies the issue of overfitting to training data set.

M5P: M5P (M5 model trees) combines a conventional decision tree with the possibility of linear regression functions at the nodes. This approach creates a compact and comprehensible model that can even predict a “class” that can have continuous numeric value.

M5Rules: Generates a decision list for regression problems using separate-and-conquer. In each iteration it builds a model tree using M5 and makes the "best" leaf into a rule.

SMO reg: Sequential minimal optimization regression is a quadratic programming problem solving algorithm, which implements the support vector machine for regression. The parameters can be learned using various algorithms.

Linear regression: This is a widely used approach to determine the relationship between response and explanatory variables. The relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the provided data

Vote: Voting is one of the ensemble classifiers, that allows to improve accuracy by means of combining several base classifiers. There are several possible ways to organize voting procedure: majority voting, average of probabilities, median of probabilities, etc.

Random committee: Random committee involves the construction of a number of base classifiers using unique random number seed values and the final classification result is provided by computing the average of the predictions generated by the individual base classifiers.

Random tree: Random tree is analogous to decision tree where a random subset of attributes is available for each branch. It can generate a classification model which is able to predict the value of the label based on several input attributes of the given dataset.

Additive regression: The additive regression model specifies that the average value of response variable is the sum of separate terms for each predictor, while excluding interaction between the predictor variables. Additive regression model can be reduced to a series of two-dimensional partial regression problems. Despite additive regression model is more restrictive than the general nonparametric regression model, it is more flexible than the standard linear regression model.

REPTree: Reduces Error Pruning (REP) Tree Classifier is a fast decision tree learning algorithm which is based on the principle of computing the information gain with entropy and minimizing the error arising from variance

Gaussian process: It utilizes a measure of the similarity between points (the kernel function) to predict the value for an unseen point from training data. The prediction is not just an estimate for that point, but also has uncertainty information.

ZeroR: ZeroR is a classification method for predicting the majority category (class) by only considering the target, while ignoring all predictors. It is mainly used for benchmarking to compare other classification methods.

Decision stump: A decision stump is a simplified version of decision tree with only one internal node that can make prediction based on the value of just a single input feature.

Bagging: Bootstrap aggregation (Bagging) is a simple but powerful ensemble method. It can be used to lower variance for algorithms that have high variance.

Stacking: tacking is another ensemble learning technique for combining multiple classification or regression models through a meta-classifier or a meta-regressor. The base level models are trained on the given training set. Afterwards, the outputs of initial base level model are taken as features, on which the meta-model is trained.

LWL: Locally Weighted Learning (LWL) is a class of function approximation techniques, where a prediction is done by using an approximated local model around the current point of interest. It can perform both classification and regression.

**Methodology, Mathematical description**

Problem parameters:

Initial inventory: I0

Inventory at the end of month t: It

Forecasted demand for product in month t: FDt

Real demand for product in month t: RDt

Ordered quantity at month t: QOt

Holding cost/month/piece: HC1 (for inventory is up to It1) = $ 1

HC2 (for inventory beyond It1) = $ 2

Amount of backorder in month t: QBt

Backorder cost/unit: BC = $ 3

Decision variable:

Ordered quantity at month t: Qt

Objective function:

Minimize (holding cost + backorder cost)

*min {HC1 It1 + HC2 It2 + QBt BC}*

Constraints:

It+1 = (QOt + It – QBt - RDt+1) + [where, X+ = max {0, X}]

QBt+1 = (RDt+1 – QOt - It)) +

QOt = FDt+1 + QBt - It

It = It1 + It2

0 ≤ It1 ≤ 90

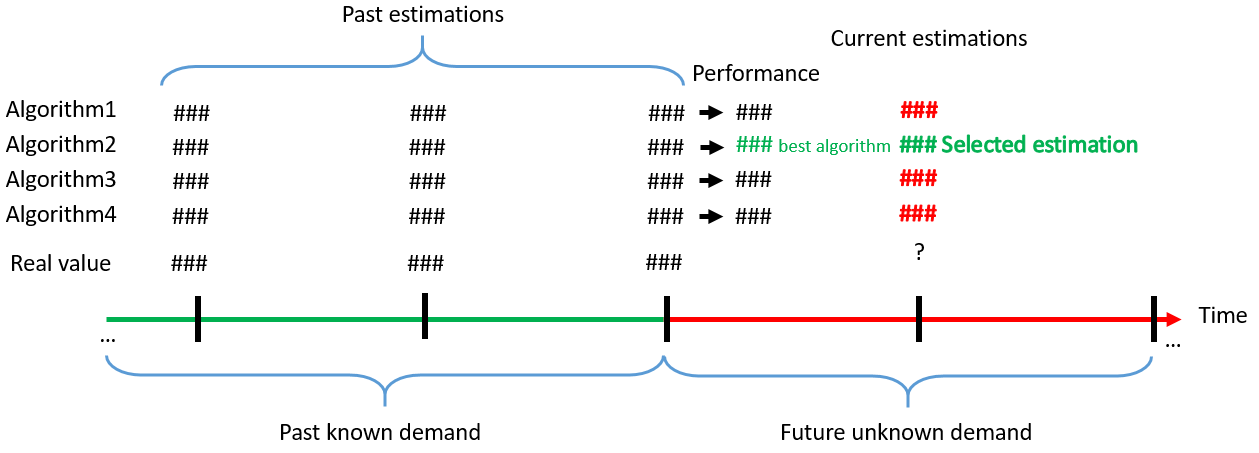
It2 = (It – 90) +

Dt, Qt, It1, It2, Bt ≥ 0

For estimating the monthly demand, Weka forecasting package[[1]](#footnote-1) and ARMA[[2]](#footnote-2) were used. By utilizing the source code of Weka package, a simulator is designed to call Weka functions on each month.

In order to get good estimations, on each month, the model is retrained and the demand for the next month is estimated. In the literature this is called Bayesian model or Active learning. We tested several algorithms and procedures. The contribution of us on this process is that, on each month all algorithms output their estimations and the performance of each algorithm is assessed by its performance on past months. We call this procedure “Active Algorithm”. Similar to ensemble learners that do bootstrap sampling, here each learner can be the chosen algorithm in a specific time interval. We hope that the winning algorithm in time t is going to outperform other algorithms for the incoming month t+1. The performance comparison of the algorithms is done by comparing the total cost calculated for each algorithm based on its estimations until now. If an algorithm fails by any means, its estimated demand will be the average of the estimations that it didn’t fail before. The estimated demand is shifted by one sigma based on its confidence interval. This value is estimated by manually checking several values and checking.

The process can be summarized in the figure below.



The algorithms can be listed in the following table.

|  |  |
| --- | --- |
| **Name** | **Description** |
| ARIMA | Autoregressive integrated moving average |
| GP | Gaussian process |
| LR | Linear regression |
| SMOreg | Sequential minimal optimization regression |
| IBk D | K-nearest neighbor on discretized space |
| LWL | Locally Weighted Learning |
| AR | Additive regression |
| Bagging+BN D | Bagging on Bayesian net on discretized space |
| Bagging+GP | Bagging on Gaussian process |
| Bagging+LR | Bagging on linear regression |
| Bagging+MP | Bagging on multilayer perceptron |
| Bagging+AdBoost(DS) D | Bagging on AdaBoostM1 with Decision stump on discretized space |
| Bagging+LB(DS) D | Bagging on Logit Boost with Decision stump on discretized space |
| Bagging+JRip D | Bagging on fast effective rule induction on discretized space |
| Bagging+OneR D | Bagging on 1R (minimum error attribute for prediction) on discretized space |
| Bagging+PART D | Bagging on PART (partial C4.5 decision tree) on discretized space |
| RC+RT | Random committee on random tree |
| RC+GP | Random committee on Gaussian Processes |
| ST+RF+GP | Stacking on random forest and Gaussian Processes |
| V+GP+LR+(Lo D) | Vote on Gaussian Processes and Linear Regression and Logistic regression on discretized space |
| V+RF+GP | Vote on Random Forest and Gaussian Processes |
| M5Rules | M5Rules (decision tree) |
| ZeroR | Majority class |
| DS | Decision Stump |
| M5P | Regression tree |
| RF | Random forest |
| RT | Random tree |
| REPTree | Decision tree |
| AdBoost D+DS | AdaBoostM1 on discretized space with Decision stump |

**Read Me**

The code is written in Java. The repository contains Netbeans 11 project. However, you can find the compiled version in “FinalDist” folder. By running the code, you will see a GUI. There is button for loading the data. The data has to be in the same format of the training data provided i.e. first column is year that is not repeated until next year, next column is month, and next column is demand. Parameters are described below:

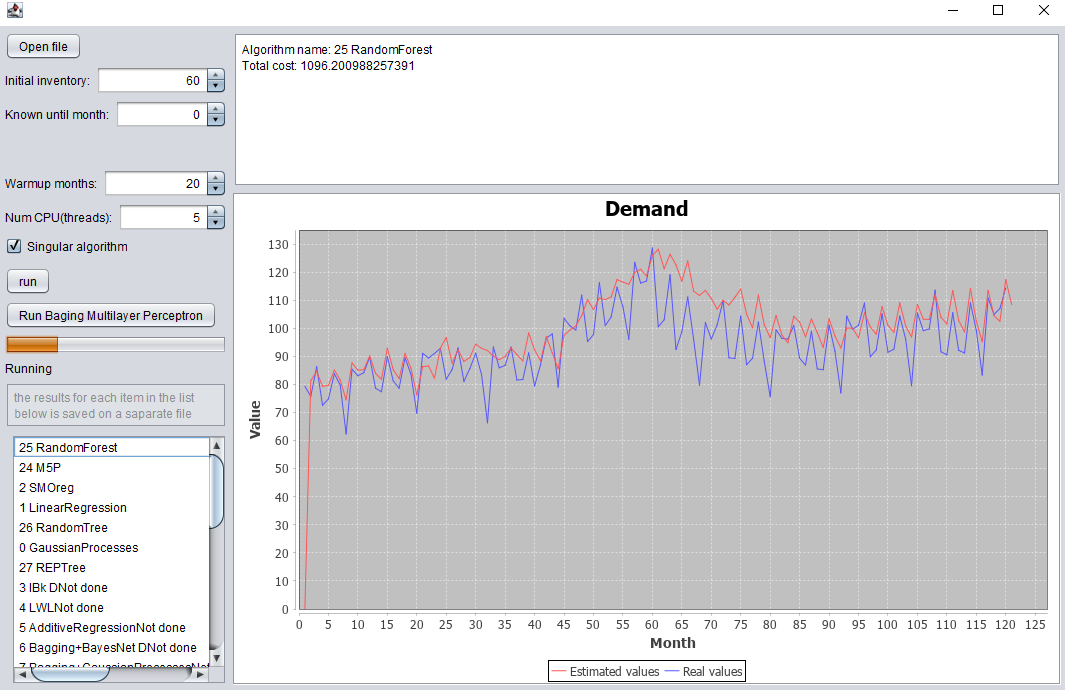
-Initial inventory: This implies the initial inventory on the month that we don’t have the demand data anymore. By default, it should be 73.

-Known until month: This parameter determines how many months were known beforehand. Total cost calculations ignore the known months. By default, it should be 120.

-Warmup months: This parameter is useful when it is intended to have “Known until month=0”. The problem is that there should be a minimum number of known months to start predictions. Therefore, this parameter should be exactly the same as “Known until month=120” because known months are also warmup months for the models.

-Num CPU(threads): This parameter is for parallel processing. If the number is set to 1, software runs in serial and if it exceeds 30, it’s redundant. The value should be around 24. Most of the algorithms run fast and the bottlenecks are ensemble learners. Therefore, setting this number more than the number of cores will not utilize all CPU power.

-Singular algorithm checkbox: This checkbox is a controller to whether do Active Algorithm method or one single algorithm.



The GUI automatically lists the best algorithms while running in single algorithm mod. To run the code, Java8 is required. This software is made by Azul Zulu OpenJDK. The final result is saved as CSV file in the current directory of application. This directory when ran from Netbeans is the root folder of the project and if ran outside is the folder that the Jar file exists.

**Result and discussion, comparison of other algorithms**

The results show that the “Active algorithm” is not outperforming single algorithms. It’s hard to assess the root cause because of the time limits for the contest. Finally, Bagging Multi-Layer Perceptron is selected as the dominant algorithm. In order to evaluate on the test data set please refer to Bagging Multi-Layer results because we assume that it should outperform other algorithms. You can run this algorithm only, by clicking on “Run Bagging Multilayer Perceptron”.

1. https://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka [↑](#footnote-ref-1)
2. https://github.com/AdairZhao/ARIMA [↑](#footnote-ref-2)