Paper 3614-2019

Data-Driven Agent Allocation in a Process using Machine Learning and Optimization in SAS®

Lokendra Kumar Devangan, Chandu Saladi, Core Compete

ABSTRACT

Data-driven agent allocation provides immense opportunity in improving the efficiency of any process. Cost effective system can be designed using machine learning and optimization in SAS®. In this paper, the application of machine learning and optimization for agent allocation in two-stage job processing is described using a case study of Business Process Outsourcing (BPO) organization. This organization handles verification and underwriting process for credit card applications of a large bank. The paper provides a brief overview of unsupervised machine learning algorithms for agent and application profiling. Clusters of application and agents are created and are used in the agent allocation optimization problem. Optimization model framework is extensively discussed to solve the problem of skill-based agent allocation for credit card application processing based on its complexities, which has two stages of processing, application verification, and underwriting. A mixed integer optimization problem is modeled and is solved using the OPTMODEL procedure. The design for the end-to-end process to implement optimization for agent allocation is also discussed in this paper.

INTRODUCTION

This paper discusses how to use unsupervised machine learning algorithms for agent and application profiling using historical data and design an optimization system for agent allocation to a credit card application process. An application can be of any type such as visa, credit card, loan, passport etc. To solve the optimization problem, PROC OPTMODEL have been used. SAS Visual Statistics has been used for profiling of agents and applications. The business problem, solution design, and assumptions are discussed in this paper. Subsequent section briefly discusses settings for agent and application profiling using clustering technique. An extensive description for skill-based agent allocation using optimization technique has been discussed in the following sections.

AGENT AND APPLICATION PROFILING

It is well known that the level of skills of different individuals varies, the same applies for the complexity of applications. In this case, agents are clustered into homogenous groups using demographic variables and profession-related information such as age, education, residence, income, tenure in the organization, total tenure, experience, etc. Clusters developed using K-Means clustering technique on SAS Visual Statistics have been profiled by analyzing time taken by agents for verification and underwriting of applications in the past. The objective of the analysis was to create a meaningful group. Based on the skill level, three clusters were created using K- Means clustering technique. The clusters were named High (H), Medium (M) and Low(L).

Similarly, credit card applications can be clustered using application attributes like applicant's age, profession, education, residence, income, new customer, credit score, documentation status, application mode, time of the year, etc. All available attributes were used to create segments using K-Means clustering on SAS Visual Statistics. Two separate clustering exercise was performed to create complexity level for two stages of the application decision process, verification and underwriting. These clusters were used to assign a complexity level of 0 to 3. Figure 1 Clustering Results shows the result window of SAS Visual Statistics.

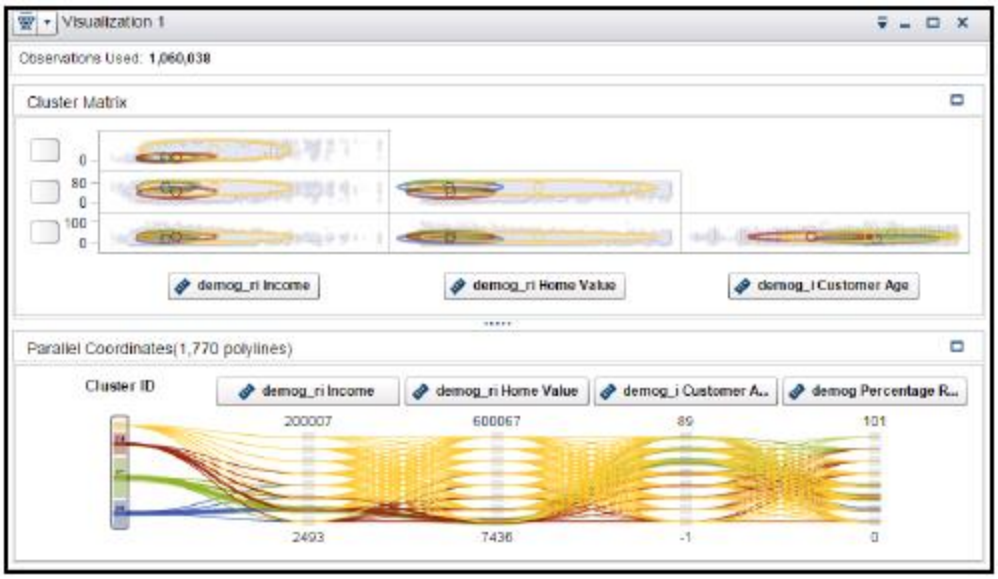

Figure 1 Clustering Results

Clusters developed for agents and applications to be grouped by skills and complexity respectively were used to create a table as illustrated in **Table 1** for each stage of the application processing. These tables form input to the optimization model and is discussed in the next section.

Table 1

Underwriting			
Skill Level	1	2	3
High	Mean time taken by High skilled agents on applications with complexity level 1		
Medium			
Low			

AGENT ALLOCATION OPTIMIZATION SETTING

In this section, settings and assumptions of the application processing have been described. Every day in the morning, all the pending applications are allocated to the agents for verification and for underwriting and they remain allocated to the same agent until a decision is made. If the agent is required to be changed, it will be initiated by a manager using other available ways rather than using the optimization system. It is assumed that for allocation, there is a system which decides whether an application requires only verification or only underwriting or both and henceforth designated complexity on a 0-3 scale with 3 being the most complex. 0 means verification is not required. All applications are required to go through underwriting though. The application included in the optimization are:

- Applications arriving in the system today which are analyzed to asses if it needs verification or underwriting or both.

- Applications from previous days that required verification or underwriting and were not allocated earlier.

Any application allocated earlier, which were not completed would not be included again. They remain allocated to the same agent and if scheduled for today, the availability of that agent is reduced for today. Agents are grouped into High/Medium/Low skill blocks for verification and underwriting. Hence, there are 6 possible resource blocks to which an application can be allocated, at most one in verification and one in underwriting. For each agent, information about previous application allocation to the agent, which has not been completed yet is assumed to be provided. In practice, this would be small since the allocation is made in such a way that the tasks will be completed on the same day. Hence, the capacity of each agent today is known, based on their availability. Aggregating across all High/Medium/Low skills across verification or underwriting, the total capacity for each of the six blocks is known. Based on the previous allocations which are not finished but due for today, the schedule for the first part is already known and hence it is not part of optimization.

DATA ORGANIZATION

Assumptions

1. A Workday is divided into $\text{TimeBlock} = 1, \ldots . t$ blocks e.g., 8 for each hour from 9.00 a.m. to 6.00 p.m. except for the lunch hour assumed from noon to 1:00 p.m. It can be grouped into fewer or more homogeneous time blocks.
2. All agents are profiled using historical data and grouped into High (H), Medium(M) and Low(L) skill level.
3. All agents skilled for verification or underwriting are grouped together and are treated as a resource block.
4. It is assumed that all agents with skillset H/M/L will take the same mean time for verification and underwriting job with complexity j/p.
5. Time availability data for each agent block by skill level in a time block t is available.
6. Completion in timeslot t means a total wait of t for each application in that slot, Wait (t) = t.
7. Weight denotes the importance of applications. Applications from high valued customer could have higher weight given by known constants Importance (π).

Notation

i: index for application $i = 1 \text{ to } n$

j: index for verification complexity where $j = 0 \text{ to } 3$ and 0 means no verification and 3 is the most complex

k: index for verification agent block skill level $k =$ H/M/L

t: index for time block

p: index for underwriting complexity where $j = 1 \text{ to } 3$ and 1 least is complex and 3 is the most complex

q: index for underwriting agent block skill level $q =$ H/M/L

μ_{jk} mean processing time for verification of application with complexity j by an agent with skill k

μ_{pq} mean processing time for underwriting for application with complexity p by an agent with skill q. If an agent group does not do the task, a high number like 10000 can be used for the meantime.

V_{kt}: Available time for verification agent block with skill level k in time block t

U_{qt}: Available time for underwriting agent block with skill level q in time block t

Decision: There are two decisions for each application taken separately for verification and underwriting; which agent block and which time slot for processing.

$\boldsymbol{x_{iat} = 1}$ if application i is assigned to block t for verification by agent block a

$\boldsymbol{y_{ibt} = 1}$ if application i is assigned to block t for underwriting by agent block b

OPTIMIZATION DESIGN

Objective:

Minimize application processing time + scaled-weighted wait time

$$\text{Minimize } \sum_i \sum_j \sum_k \sum_t x_{ijkt} * W_t * \pi_i * 60 + \sum_i \sum_p \sum_q \sum_t y_{ipqt} * W_t * \pi_i * 60 + \sum_i \sum_j \sum_k \sum_t x_{ijkt} * \mu_{jk} + \sum_i \sum_p \sum_q \sum_t y_{ipqt} * \mu_{pq}$$

Binary Constraint:

$\sum_j \sum_k \sum_t x_{ijkt} = v_i \ \ for\ all\ i$ Assign to one agent and time slot for verification of application

$\sum_j \sum_k \sum_t y_{ipqt} = u_i \ \ for\ i$ Assign to one agent and time slot for underwriting of application

where v_i and u_i are vectors of 1 or 0 based on whether an application needs verification/underwriting (1) or not (0). 0 is not applicable for underwriting. This helps to specify the constraint efficiently.

Precedence Constraint:

The verification process must precede the underwriting process. This is implemented using logical constraints which ascertains that the waiting time of an application for the verification process should be less than waiting time for underwriting

$$\sum_j \sum_k \sum_t W_t * x_{ijkt} \leq \sum_j \sum_k \sum_t W_t * y_{ipqt} + (1 - u_i) * xLarge$$

Processing Time Constraint:

Total processing time should not exceed the available time in each time block for agent block with a certain skill level. For example, for all H skill verification agents in time slot 1, the time taken by all applications assigned to them should be less than the time available.

$$\sum_i \sum_j x_{ijkt} * \mu_{jk} \leq V_{kt}$$

$$\sum_i \sum_p y_{ipqt} * \mu_{pq} \leq U_{qt}$$

There can be many extensions to the mathematical formulation. One of the extensions is, if the resource capacity is less than the demand, a resource block can be added each for verification or underwriting with a very large capacity so that allocating to that block would mean that the demand will not be met that day. The mean time for this block would be high so that it is not selected except to ensure feasibility of the capacity constraint.

PROC OPTMODEL IMPLEMENTATION AND RESULTS

Proc OPTMODEL has been used to implement the mathematical model described in the previous section. Data discussed in section 2 and 3 have been transformed to ensure that it can be used as input data to Proc OPTMODEL.

```
Proc optmodel FDIGITS=3 printlevel=3;
*** declare dimensions to read the parameter data

   set <num,num> compl_dimv;
   set <num,num> compl_dimm;
   set <num> app_dim;
   set <num,num> agent_dim_v;
   set <num,num> agent_dim_m;
   set <num,num> avail_hrs_v;
   set <num,num> avail_hrs_m;
   set <num>time_slot_dim;

   /*****Declaring the parameters***********/
   num avail_time_v{avail_hrs_v};
   num avail_time_m{avail_hrs_m};
   num avgtime_v {agent_dim_v};
   num avgtime_m {agent_dim_m};
   num wait {time_slot_dim};
   num v_flag{app_dim};
   num m_flag{app_dim};
   num imp{app_dim};

/***Reading the parameters from the SAS data into Proc Opt Model*/
Read data &avail_time_v into avail_hrs_v=[k t] avail_time_v = avail_time;
Read data &avail_time_m into avail_hrs_m=[q t] avail_time_m=avail_time;

Read data &time_slot_wait into time_slot_dim=[t] wait;
Read data &m_skill into agent_dim_m=[q p] avgtime_m;
/***Mean processing time of underwriting by agent with a skill and
application complexity***/

Read data &v_skill into agent_dim_v=[k j] avgtime_v;
/***Mean processing time of verification by agent skill and application
complexity***/

Read data &applications  into compl_dimv=[i j] ;
Read data &applications  into compl_dimm=[i p] ;
Read data &applications  into app_dim=[i] v_flag imp ;
Read data &applications  into app_dim=[i] m_flag;

  /******Declaring the decision variables**********/
var slotv  {compl_dimv cross avail_hrs_v} >= 0 <=1  integer;
var slotm  {compl_dimm cross  avail_hrs_m} >= 0 <=1 integer;
```

```
/***minimize the total wait time due to verification + wait time due to
underwriting + processing time due to verification + processing time due to
underwriting* balanced by a factor**********/

minimize time = sum {<i, j,k, t> in compl_dimv  cross avail_hrs_v  :< t> in
time_slot_dim }
slotv[i,j, k, t]*wait[t]*60*imp[i]*1/&Bal_factor.+   /*wait time due to
verification **multiplying with 60 to convert to minutes**/

sum{<i, p, q, t> in compl_dimm cross avail_hrs_m  :<t> in time_slot_dim  }
slotm[i, p,q, t]*wait[t]*60*imp[i]*1/&Bal_factor. +     /*wait time due to
underwriting**multiply with 60 to convert into minutes**/

sum{<i,j,k,t> in compl_dimv cross avail_hrs_v:<k,j> in agent_dim_v }
slotv[i,j,k,t]*avgtime_v[k,j]  + /***Processing time for
verification***********/

sum{<i,p,q,t> in compl_dimm cross avail_hrs_m :<q,p> in agent_dim_m }
slotm[i,p, q, t]*avgtime_m[q,p]   ;     /***Processing time for
underwriting***********/

/*****assignment constraint for verification***/

con assignment_v { <i> in app_dim }:
sum {< (i),j,k, t> in  compl_dimv cross avail_hrs_v}
slotv[i,j, k, t]=v_flag[i];

/**** assignment constraint for  underwriting***********/
con assignment_m { <i> in app_dim}:
sum {< (i),p,q, t> in  compl_dimm cross avail_hrs_m}
slotm[i,p, q, t]=m_flag[i];

/***Processing_time constraint */
/****Total time for verification/underwriting processing by agent block by
skill level should be less than available time of the
verification/underwriting agent block***/
con agent_v {<k,t> in avail_hrs_v}:
sum {< i,j,(k),(t)> in  compl_dimv cross avail_hrs_v } slotv[i,j, k,
t]*avgtime_v[k,j] <= avail_time_v[k,t];

con agent_m {<q,t> in avail_hrs_m}:
sum {<i,p,(q),(t)> in  compl_dimm cross avail_hrs_m } slotm[i,p, q,
t]*avgtime_m[q,p] <= avail_time_m[q,t];

/*Precedence Constraints: verification must precede underwriting*/

con precedence_cons { <i> in app_dim}:
sum{<(i),j,k,t> in  compl_dimv cross avail_hrs_v:<t> in time_slot_dim}
slotv[i,j, k, t]*wait[t]  <=
sum{<(i),p,q, t> in  compl_dimm cross avail_hrs_m:<t> in time_slot_dim}
slotm[i,p, q, t]*wait[t]+ (1-m_flag[i])*10000;

solve with MILP;

/****creating SAS table to retain the solution ***********/
create data verification
```

```
from [i j k t ]
={<i,j, k, t> in compl_dimv cross avail_hrs_v :
slotv[i,j, k, t].sol > 0}
slotv = slotv;

create data underwriting
from [i p q t ]
={<i,p, q, t> in compl_dimm cross avail_hrs_m :
slotm[i,p, q, t].sol > 0}
slotm = slotm;
Quit;
```

ILLUSTRATIVE EXAMPLE

The above-formulated problem was solved for 1,104 applications with three levels of complexities for both verification and underwriting stages. Agents have been grouped as two types of resource blocks for verification and underwriting having three levels of skills. 39 applications do not require verification. 240 minutes were allocated to each resource block in a time slot for each skill level. Example problem used is considerably large with 3,360 constraints and solved using Branch and Cut algorithm. Figure 2 is screenshot of the ODS (Output Delivery System) output of Proc OPTMODEL.

Problem Summary	
Objective Sense	Minimization
Objective Function	Time
Objective Type	Linear
Number of Variables	52992
Bounded Above	0
Bounded Below	0
Bounded Below and Above	52992
Free	0
Fixed	0
Binary	52992
Integer	0
Number of Constraints	3360
Linear LE (<=)	1152
Linear EQ (=)	2208
Linear GE (>=)	0
Linear Range	0
Constraint Coefficients	158976

Solution Summary	
Solver	MILP
Algorithm	Branch and Cut
Objective Function	Time
Solution Status	Time Limit Reached
Objective Value	8181.0168353
Relative Gap	0.1840644426
Absolute Gap	1271.7502945
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	6909.2665408
Nodes	324
Solutions Found	4
Iterations	48932
Presolve Time	1.00
Solution Time	499.33

Figure 2 Problem and Solution Summary

Table 2 and Table 3 below summarizes the application allocation to agents by complexity type for illustrative example.

- Table 2 and Table 3 suggests that optimization model allocates more number of appications to high skilled agent resource block as objective is to minimize the waiting time to make a decision on application.
- It is also observed that high skilled agent resource blocks are allocated significantly large number of high complexity applications.
- Table 3 suggests that low skill agent resource block of underwriting is not allocated any application. Minimum utilization constraints can be introduced to avoid this situation.

Availability of a resource block is 240 minutes for each time block, uniform across the skill levels. In reality availability of high skill agents would not be same as low skill agents as it may not be cost effective. Power of optimization model can realized by solving illustrative example for different scenarios.

Table 2

Verification Agent Block	Verification Complexity	Number of Application
High	3	167
High	2	119
High	1	410
Medium	3	72
Medium	2	19
Medium	1	71
Low	3	23
Low	2	19
Low	1	165

Table 3

Underwriting Agent Block	Underwriting Complexity	Number of Application
High	3	268
High	2	126
High	1	265
Medium	3	178
Medium	2	86
Medium	1	181

CONCLUSION

SAS Visual Statistics is useful for developing supervised and unsupervised machine learning models quickly on large data sets as data resides on LASR. The Modeler can tune the available parameters to generate the best models. It also provides different ways of analyzing the results. Scoring codes can be exported outside for a batch run. PROC OPTMODEL is a very powerful and flexible programming language for solving different types of optimization problem. Programming in PROC OPTMODEL has the same format as the mathematical representation of optimization problems.

In the service industry, service operations play a critical role in the business. If the operations are not handled efficiently, it can lead to additional cost and impact customer satisfaction adversely. Processing time can be reduced by optimizing resource allocation based on the

skills. In this paper, a two-stage process is used to demonstrate the power of optimization and the capability of Proc OPTMODEL. Numerical results have been presented to illustrate the applicability of optimization in the service industry to optimize resource allocation and processes.

REFERENCES

SAS Institute Inc. 2011. SAS/OR® 9.3 User's Guide: Mathematical Programming. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Special thanks to Dr. Ajay Mishra, Core Compete for the guidance provided for the optimization model development.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Lokendra Kumar Devangan
Core Compete
E-mail: Lokendra.devangan@corecompete.com
www.corecompete.com

Chandu Saladi
Core Compete
E-mail: Chandu.Saladi@corecompete.com
www.corecompete.com