Garment Industry: Predicting Productivity

Using Regression and Classification Models

Ángel Silvar

California State University - Long Beach

May 1, 2022

1 Introduction

This project aims to determine productivity given multiple characteristics such as idle time, incentive, number of workers. The main response variable, "actual productivity", is an element within the unit interval leading us to mutate it into a binary variable named "productivity".

Name	Description
Date	Date
Day	Day of the week
quarter	A month was divided into four quarters
department	Associated department: finishing
team	Associated team number 1-12
no of worker	Number of workers in each team
no of style_change	Number of changes in the style of a particular product
targeted productivity	set by the Authority for each team for each day
smv : Standard Minute Value	it is the allocated time for a task
wip : Work in progress	Includes the number of unfinished items for products
over time	Represents the amount of overtime by each team in minutes
incentive	Represents the amount of financial incentive
idle time	Time production was interrupted due to several reasons
idle men	Number of workers in idle due to production interruption
actual productivity	the actual productivity by percent
productivity	"Yes", if 80% completion is reached

Table 1 – Attributes of the Dataset.

2 Research Questions

Both Regression and Classification models were applied using 14 predictors to find productivity continuously or as a binary. Our research questions:

- Can we find a model which most accurately predicts productivity given the 14 predictors?
- How do incentives correlate with productivity?
- Does the idle time affect productivity?
- What is the predicted productivity for large idle time and idle men?

3 Analysis

3.1 Data Preparation

The data set includes nearly 1200 observations along with 16 variables. The data set was opened through Google Sheets to observe the data set. It was noticed that variable wip has empty elements. Dropping NAs gives us a subset with about 700 observations with 15 variables where we lose department variable. Note, that variable, day, was dropped as it is already covered by variable "quarter".

It was mentioned that targeted productivity is correlated with response variable "actual productivity" . Due to this correlation, targeted variable will be dropped. See the following figure for a corelation matrix.

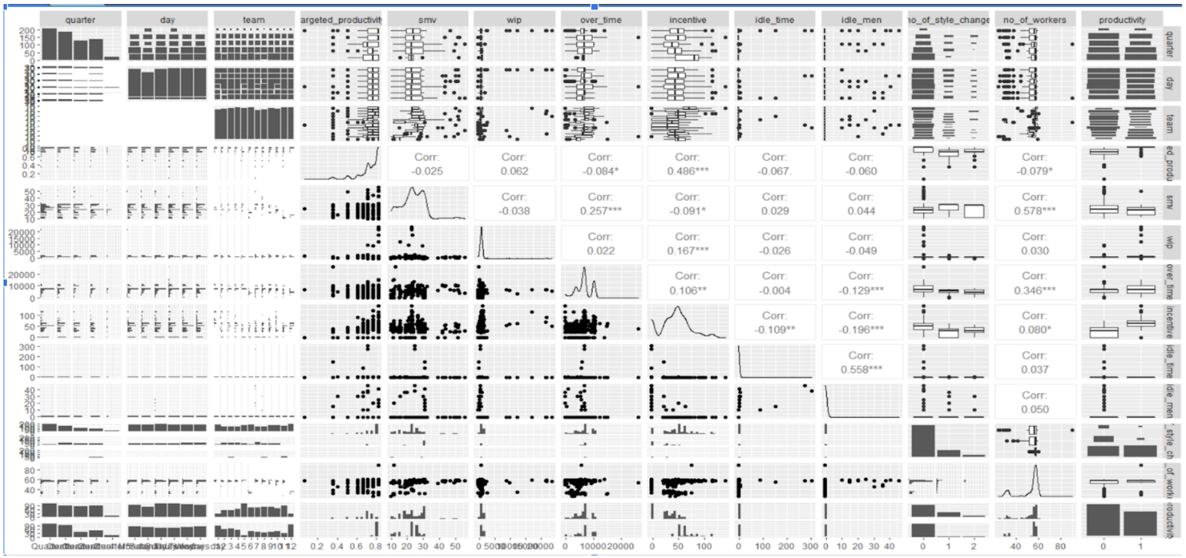
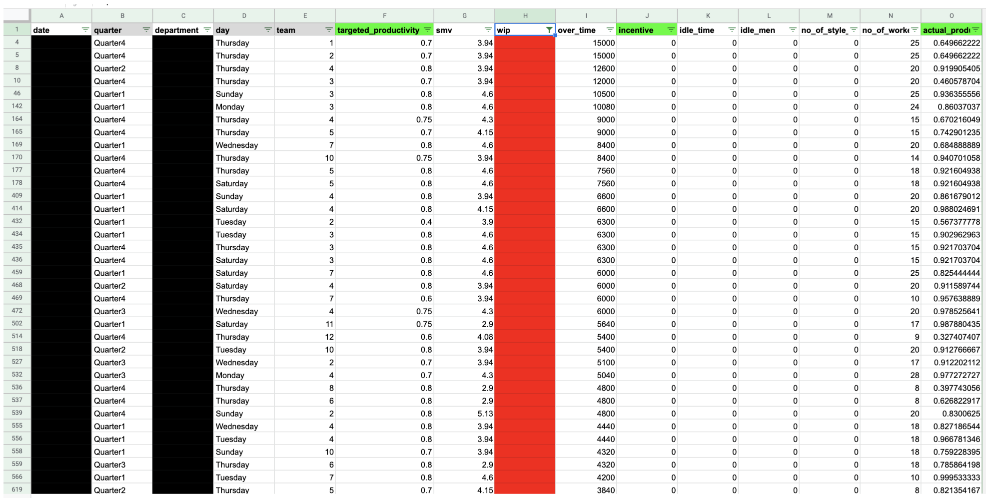


Figure 1 – Correlation Scatterplot for all values in subset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	date	quarter	department	day	team	targeted_productivity	smv	wip	over_time	incentive	idle_time	idle_men	no_of_style_	no_of_work	actual_prod
4	[redacted]	Quarter4	[redacted]	Thursday	1	0.7	3.94		15000	0	0	0	0	25	0.649662222
5	[redacted]	Quarter4	[redacted]	Thursday	2	0.7	3.94		15000	0	0	0	0	25	0.649662222
8	[redacted]	Quarter2	[redacted]	Thursday	4	0.8	3.94		12600	0	0	0	0	20	0.919905405
10	[redacted]	Quarter4	[redacted]	Thursday	3	0.7	3.94		12000	0	0	0	0	20	0.460578704
46	[redacted]	Quarter1	[redacted]	Sunday	3	0.8	4.6		10500	0	0	0	0	25	0.936355556
142	[redacted]	Quarter1	[redacted]	Monday	3	0.8	4.6		10080	0	0	0	0	24	0.86037037
164	[redacted]	Quarter4	[redacted]	Thursday	4	0.75	4.3		9000	0	0	0	0	15	0.670216049
165	[redacted]	Quarter4	[redacted]	Thursday	5	0.7	4.15		9000	0	0	0	0	15	0.742901235
169	[redacted]	Quarter1	[redacted]	Wednesday	7	0.8	4.6		8400	0	0	0	0	20	0.684888889
170	[redacted]	Quarter4	[redacted]	Thursday	10	0.75	3.94		8400	0	0	0	0	14	0.940701058
177	[redacted]	Quarter4	[redacted]	Thursday	5	0.8	4.6		7560	0	0	0	0	18	0.921604938
178	[redacted]	Quarter4	[redacted]	Saturday	5	0.8	4.6		7560	0	0	0	0	18	0.921604938
409	[redacted]	Quarter1	[redacted]	Sunday	4	0.8	3.94		6600	0	0	0	0	20	0.861679012
414	[redacted]	Quarter1	[redacted]	Saturday	4	0.8	4.15		6600	0	0	0	0	20	0.988024691
432	[redacted]	Quarter1	[redacted]	Tuesday	2	0.4	3.9		6300	0	0	0	0	15	0.567377778
434	[redacted]	Quarter1	[redacted]	Tuesday	3	0.8	4.6		6300	0	0	0	0	15	0.902962963
435	[redacted]	Quarter4	[redacted]	Thursday	3	0.8	4.6		6300	0	0	0	0	15	0.921703704
436	[redacted]	Quarter4	[redacted]	Saturday	3	0.8	4.6		6300	0	0	0	0	15	0.921703704
459	[redacted]	Quarter1	[redacted]	Saturday	7	0.8	4.6		6000	0	0	0	0	25	0.825444444
468	[redacted]	Quarter2	[redacted]	Saturday	4	0.8	3.94		6000	0	0	0	0	20	0.911589744
469	[redacted]	Quarter4	[redacted]	Thursday	7	0.6	3.94		6000	0	0	0	0	10	0.957638889
472	[redacted]	Quarter3	[redacted]	Wednesday	4	0.75	4.3		6000	0	0	0	0	20	0.978525641
502	[redacted]	Quarter1	[redacted]	Saturday	11	0.75	2.9		5640	0	0	0	0	17	0.987880435
514	[redacted]	Quarter4	[redacted]	Thursday	12	0.6	4.08		5400	0	0	0	0	9	0.327407407
518	[redacted]	Quarter2	[redacted]	Tuesday	10	0.8	3.94		5400	0	0	0	0	20	0.912766667
527	[redacted]	Quarter3	[redacted]	Wednesday	2	0.7	3.94		5100	0	0	0	0	17	0.912202112
532	[redacted]	Quarter3	[redacted]	Monday	4	0.7	4.3		5040	0	0	0	0	28	0.977272727
536	[redacted]	Quarter4	[redacted]	Thursday	8	0.8	2.9		4800	0	0	0	0	8	0.397743056
537	[redacted]	Quarter4	[redacted]	Thursday	6	0.8	2.9		4800	0	0	0	0	8	0.626822917
539	[redacted]	Quarter4	[redacted]	Sunday	2	0.8	5.13		4800	0	0	0	0	20	0.8300625
555	[redacted]	Quarter1	[redacted]	Wednesday	4	0.8	3.94		4440	0	0	0	0	18	0.827186544
556	[redacted]	Quarter1	[redacted]	Tuesday	4	0.8	3.94		4440	0	0	0	0	18	0.966781346
558	[redacted]	Quarter1	[redacted]	Sunday	10	0.7	3.94		4320	0	0	0	0	18	0.759228395
559	[redacted]	Quarter3	[redacted]	Thursday	6	0.8	2.9		4320	0	0	0	0	18	0.785864198
566	[redacted]	Quarter1	[redacted]	Tuesday	7	0.8	4.6		4200	0	0	0	0	10	0.999533333
619	[redacted]	Quarter2	[redacted]	Thursday	5	0.7	4.15		3840	0	0	0	0	8	0.821354167

Figure 2 – Google Sheets with depicting NA's.

3.2 Logistic Regression

We know that actual productivity is an element of the unit interval. Our goal is to run classification models by mutating actual productivity into new variable. Here the threshold for new variable, productivity, to be successful was if actual productivity is greater than .80.

Run the logistic model. Then, rerun Logit model on variables that meet 95% threshold.

```
Call:
glm(formula = productivity ~ ., family = "binomial", data = train_set2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1625  -0.4182  -0.0363   0.4001   3.5824

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -7.768e+00  2.393e+00  -3.246  0.00117 **
quarterQuarter2      4.256e-01  4.009e-01   1.062  0.28838
quarterQuarter3     -3.503e-01  4.228e-01  -0.829  0.40737
quarterQuarter4     -6.727e-01  4.332e-01  -1.553  0.12048
quarterQuarter5     -1.451e+00  1.029e+00  -1.410  0.15857
daySaturday          4.859e-02  4.975e-01   0.098  0.92220
daySunday           -3.295e-01  4.994e-01  -0.660  0.50937
dayThursday          5.118e-02  5.081e-01   0.101  0.91976
dayTuesday          -2.488e-01  4.582e-01  -0.543  0.58711
dayWednesday         3.270e-02  4.573e-01   0.071  0.94300
team2                7.160e-01  8.703e-01   0.823  0.41062
team3               -1.250e-02  7.445e-01  -0.017  0.98661
team4                1.462e-01  7.303e-01   0.200  0.84134
team5               -1.179e+00  8.947e-01  -1.318  0.18766
team6                1.703e+00  1.208e+00   1.409  0.15873
team7                1.866e+00  8.222e-01   2.269  0.02324 *
team8                8.135e-01  7.973e-01   1.020  0.30755
team9               -7.400e-01  7.281e-01  -1.016  0.30943
team10              -6.008e-01  8.003e-01  -0.751  0.45277
team11              -1.401e+00  8.427e-01  -1.662  0.09649 .
team12               2.273e+00  1.215e+00   1.871  0.06137 .
smv                 -9.496e-02  3.493e-02  -2.718  0.00656 **
wip                 -3.613e-05  8.062e-05  -0.448  0.65403
over_time           -1.769e-04  6.452e-05  -2.742  0.00611 **
incentive            1.267e-01  1.365e-02   9.280  < 2e-16 ***
idle_time            3.159e-02  1.680e+01   0.002  0.99850
idle_men            -1.228e+00  6.511e+01  -0.019  0.98495
no_of_style_change1 -1.374e+00  5.633e-01  -2.439  0.01472 *
no_of_style_change2 -1.126e+00  8.647e-01  -1.302  0.19280
no_of_workers        9.757e-02  4.322e-02   2.258  0.02398 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 754.74  on 551  degrees of freedom
Residual deviance: 334.69  on 522  degrees of freedom
AIC: 394.69

Number of Fisher Scoring iterations: 17
```

Figure 3

3.3 Linear Discriminate Analysis: A Different Approach to Classification

The binary variable, prediction, is successful when 80% of actual productivity is reached. This partitions the data set somewhat evenly where $productivity = 44.44\%$ of the time. In other words, out of 693 observations, $productivity = 1,304$ times. We know that LDA is more stable when the classifiers are evenly (linearly) split. So the goal of using LDA is to see if LDA can successfully predict productivity knowing that it is more stable than Logistic Regression
The following code is meant to apply LDA model onto the data set and verify if LDA performances better by demonstrating higher stability when predicting.

This code this mean to output a confusion matrix and determine model accuracy using ROC curve and AUC.

Recall:

$$\text{Sensitivity (True Positive Rate)} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{142}{15+142} = 0.9044586 = 90.44\%$$

Hence, the our positive rate of our model is very high in predicting productivity.

```
Call:
glm(formula = productivity ~ team + over_time + smv + incentive +
    no_of_style_change + no_of_workers, family = "binomial",
    data = train_set2)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1601  -0.4304  -0.0488   0.4638   3.6452

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -7.045e+00  2.237e+00  -3.150  0.00163 **
team2                6.977e-01  8.424e-01   0.828  0.40759
team3                3.006e-02  7.241e-01   0.042  0.96688
team4                3.629e-02  7.133e-01   0.051  0.95943
team5               -1.602e+00  8.912e-01  -1.797  0.07232 .
team6                1.260e+00  1.147e+00   1.098  0.27206
team7                1.606e+00  7.847e-01   2.047  0.04067 *
team8                6.193e-01  7.562e-01   0.819  0.41283
team9               -7.912e-01  7.098e-01  -1.115  0.26500
team10              -7.664e-01  8.052e-01  -0.952  0.34122
team11              -1.505e+00  8.084e-01  -1.861  0.06271 .
team12               1.741e+00  1.146e+00   1.519  0.12868
over_time           -1.482e-04  5.676e-05  -2.611  0.00903 **
smv                 -9.352e-02  3.333e-02  -2.806  0.00502 **
incentive            1.221e-01  1.295e-02   9.428  < 2e-16 ***
no_of_style_change1 -1.329e+00  5.004e-01  -2.655  0.00793 **
no_of_style_change2 -1.490e+00  7.700e-01  -1.935  0.05294 .
no_of_workers        8.168e-02  4.071e-02   2.006  0.04480 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 754.74  on 551  degrees of freedom
Residual deviance: 351.76  on 534  degrees of freedom
AIC: 387.76

Number of Fisher Scoring iterations: 6

[1] 0.8705036
```

Figure 4

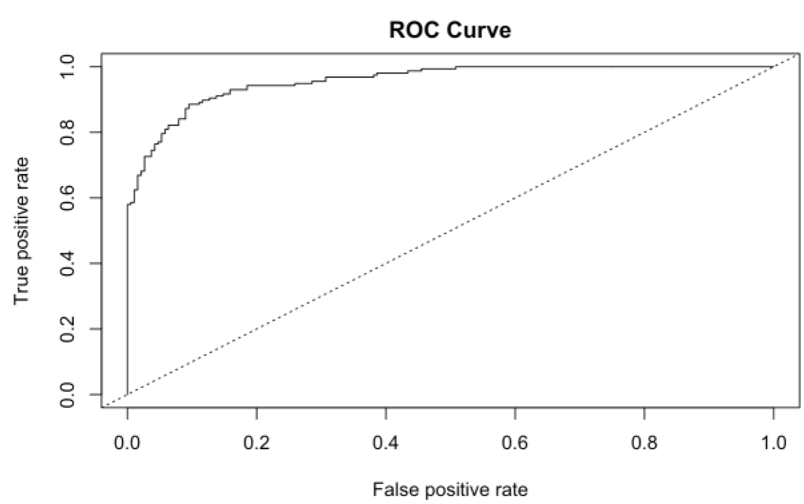


Figure 5 – ROC Curve. The desired result is for the curve to close to the upper left corner as much as possible.

```
Call:
lda(productivity ~ ., data = train_set2)

Prior probabilities of groups:
        0         1
0.5688406 0.4311594
```

Figure 6 – It shows that it was found that productivity was successful 43% of the time.

```
                true_status
predict_status  0  1
             0 66 11
             1  7 55
[1] 0.8705036
[1] 0.9445828
```

Figure 7 – Confusion matrix for LDA + Accuracy. AUC = 94%

$$\text{Specificity (True Negative Rate)} \ = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}} = \frac{66}{66+27} = 0.70967770.96\%$$

Specificity at 71% level shows that this model is also accurate enough to predict productivity incorrectly. In other words, False negative rate is at 29%

$$\text{Accuracy (Prediction Correctly Identified)} = \frac{\text{True Negative} + \text{True Positive}}{\text{True Negative} + \text{False Positive} + \ \text{True Positive} + \ \text{False Negative}}$$

$$= \frac{66+55}{66+55+7+11} = 0.87050 = 87.70\%$$

3.4 Shrinkage Method: Lasso Model

Shrinkage method is a method to improve regression and logistic models. Here, the method will take 14 predictors and shrink them by using λ parameter. Our sample size is more than 600, this helps minimize model variance but minimally increase bias. Note: since variables within the data set are not related with the response variable, Ridge regression will not be used. Instead, this section aims to seek results that come from the Lasso Model.

Our first step is to standardize our variables which means that each observation of each column will be divided by its respective standard deviation. This is presented in the following:

After applying cross validation methods, our λ is found to be 0.0037. With out MSE = 0.5. Here, we can see that variables are sent to 0 as they were found to be insignificant in predicting response variable production. With out MSE being near 0, we can use the coefficients from each variable and report our new classification model.

```
[1] 0.01041201
30 x 1 sparse Matrix of class "dgCMatrix"
                               s1
(Intercept)         -3.568742888
quarterQuarter2      0.160149355
quarterQuarter3     -0.054777551
quarterQuarter4     -0.154317169
quarterQuarter5     -0.038246060
daySaturday          .
daySunday            .
dayThursday          .
dayTuesday           .
dayWednesday         .
team2                .
team3                .
team4                .
team5               -0.269801122
team6                .
team7                0.266593225
team8                0.007085545
team9               -0.121031380
team10              -0.073313077
team11              -0.263239897
team12               0.159081094
smv                 -0.846678348
wip                  .
over_time           -0.337453633
incentive            5.136383071
idle_time            .
idle_men            -0.015199253
no_of_style_change1 -0.268963310
no_of_style_change2 -0.063617866
no_of_workers        .
[1] 0.8920863
```

Figure 8 – AUC = .95 while Accuracy is at 88%

3.5 Tree Model: A Search For Minimum requirements for Maximum Out for Prediction

The aim of this model is to understand what variables, at a specific level, yields a success in productivity for the day. A classification tree model was used to answer the following questions: what variables are the most significant, and at what value of these variables yields successful productivity

3.5.1 Classification Tree: What combination of features yields 80% level of productivity?

First comes data preparation. Here, productivity is converted to class productivity as classification tree model does not accept binary numeric values. The following tree is without pruning.

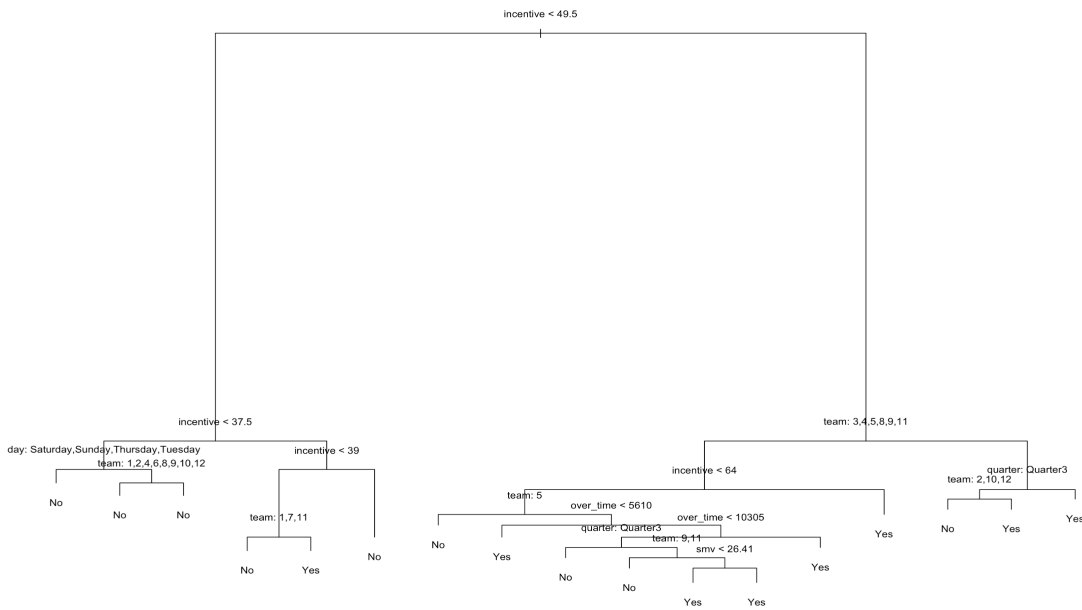


Figure 9 – Incentive dominated the decision tree of the data set.

For the best results, we prune tree model.

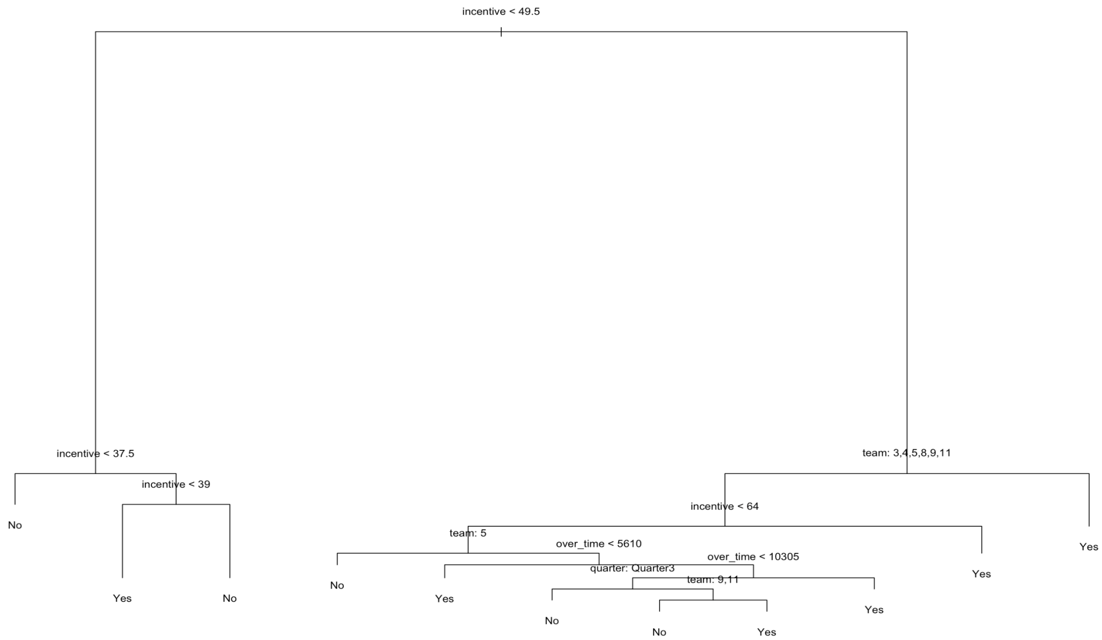


Figure 10 – Target Productivity and Incentive are the most important predictors of the data set.

The confusion matrix:

```
             true_status
predict_status No Yes
           No  67   6
           Yes  6  60
[1] 0.9136691
             true_status
predict_status No Yes
           No  67   4
           Yes  6  62
[1] 0.9280576
```

Figure 11 – Accuracy of full tree vs. Accuracy of pruned tree, respectively.

The output shows that given high target productivity and incentive lowered at 36 BDT (local currency) is the threshold that determines whether 80% of the daily qouta will be met. If incentive is higher, then it will increase chances that the quota will be met, but it may not need as much of an incentive. On the other hand, if there is a lower amount of target productivity, then it is shown that it is required much more of an incentive for workers in order to reach a 80% of the daily quota.

3.6 Extension of Tree Models: Random Forest, Bagging, and Gradient Boosting

3.6.1 Random Forest:

```
Call:
 randomForest(formula = productivity ~ ., data = train_set, mtry = round(sqrt(p)),      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 7.43%
Confusion matrix:
    0   1 class.error
0 290  24  0.07643312
1  17 221  0.07142857
```

Figure 12 – Variation Explained is at 84.27%.

```
      true
pred  0  1
   0 69  3
   1  4 63
[1] 0.9496403
```

Figure 13 – Variation Explained: 94.96% .

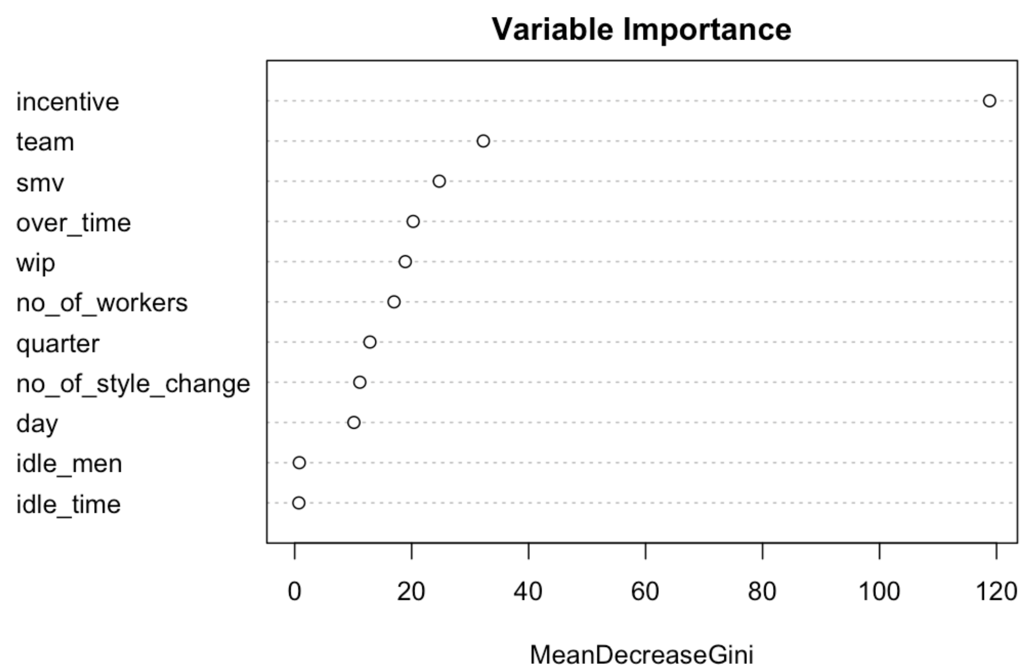


Figure 14 – Incentive is proven to be the most significant out of the bunch.

3.6.2 Bagging Model

The output of the model shows that number of tree used is 500. Mean squared residual is 0.004

```
Call:
 randomForest(formula = productivity ~ ., data = train_set, mtry = p,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 11

        OOB estimate of  error rate: 7.61%
Confusion matrix:
    0   1 class.error
0 289  25  0.07961783
1  17 221  0.07142857
```

Figure 15 – 11 nodes used along with Out of Bag Estimation.

```
                     true_status
predicted_status  0  1
                0 66  4
                1  7 62
[1] 0.9208633
```

Figure 16 – Accuracy = 92%.

3.6.3 Gradient Boosting

The following output is the confusion matrix from the pre-tuned gradient boosting model.

```
    true
pred   0   1
   0 179   7
   1  10 150
```

Figure 17 – Variation Explained: 82.94% .

Tune parameters for boosting model.

```
    true
pred   0   1
   0 180   5
   1   9 152
[1] 0.9595376
```

Figure 18 – Variation Explained: 82.94% .

3.7 Support Vector Machine

CV-10 validation was applied to all 3 variations of SVM.

3.7.1 Linear

Cost was found to be 0.1. Our support vectors are 164. Where 83 are for output being 0 and 81 are for outputs being 1.

```
      truth
pred  0  1
   0 67 12
   1  6 54
[1] 0.8705036
```

Figure 19 – Variation Explained: 82.94% .

3.7.2 Polynomial

```
      truth
pred  0  1
   0 73 57
   1  0  9
[1] 0.5899281
```

Figure 20 – Variation Explained: 82.94% .

Here our cost was found to be 10. The fit model showed that our support vector total is 105 where 46 is for outputs being 0. Our polynomial was of degree 3.

3.7.3 Radial

Here the cost was also 10. The fit moel was set to 552 vectors where 238 vectors were set to for outputs being 0.

```
      truth
pred  0  1
   0 31 13
   1 42 53
[1] 0.6043165
```

Figure 21 – Variation Explained: 82.94% .

4 Prediction: Random Forest

We concluded that random forest and gradient boosting are our best models for accuracy. Since we care more about predicting accuracy levels of determining fails and success, we decided to predict using random forest and as our main model for this task. In the interest of time, we will only using random forest as our main model. The following are predictions regarding variables incentive, over time, idle men, and idle time. We created a new data frame that considered a values of the average, minimum, maximum, and average - standard deviation, and average + standard deviation. Each are of simulation that ranges from 1-5 respectively.

4.1 Incentives

```
1 2 3 4 5
0 0 1 0 1
```

Figure 22 – Simulation 3 yielded a success along with Simulation 5.

Incentives prove to be significant when it come to predicting productivity. At our maximum value and a standard deviation above the average, yields a successful day. This shows that there is an interval between these two values (that does not include the mean) has an affect on productivity.

4.2 Over Time

```
1 2 3 4 5
0 0 0 0 0
Levels: 0 1
```

Figure 23 – Simulations were unsuccessful .

In addition, over time shows a similar characteristic with idle time and idle men. It shows, even at a maximum amount of over time, does not yield a successful day.

4.3 Idle Time

Analysis is shown in Idle Time and Idle Men.

```
1 2 3 4 5
0 0 0 0 0
Levels: 0 1
```

Figure 24 – Simulations were unsuccessful .

4.4 Idle Men

Analysis is shown in Idle Time and Idle Men.

```
1 2 3 4 5
0 0 0 0 0
Levels: 0 1
```

Figure 25 – Simulations were unsuccessful .

4.5 Idle Men and Idle Time

```
1 2 3 4 5
0 0 0 0 0
Levels: 0 1
```

Figure 26 – Simulations were unsuccessful .

Idle time and Idle men, which is time wasted due to interruptions in productivity, even at minimum values (0), does not change the outcome of having a successful day. This shows that idle time and idle men, even when the warehouse is having the worst day or best day, will always yield an unproductive day. This shows that the idle time and men are not important enough to sway the probability.

4.6 Over Time and Incentives

```
1 2 3 4 5
0 0 1 0 1
Levels: 0 1
```

Figure 27 – Simulations were successful for simulation 3 and 5 .

Combining Incentives with Over time yields a successful day. Here, it is not surprising that simulation 3 and 5 are successful as we just saw that incentives alone makes simulation 3 and 5 successful.

4.7 Conclusion

These prediction tables directly answers our research questions. Ultimately, we have found that incentives is the most important predictor in our data set and random forest is our best model behind gradient boosting. If we have more time, we would have predicted using gradient boosting. We decided away from gradient boosting as it took longer to run as our trees increased.

4.8 Appendix

````

---
title: "Final Project"
author: "Angel Silvar, Matthew Michael Rublee, Safwat Abdul Rahman"
date: "4/21/2022"
output: html_document
---

```{r include=FALSE ,message=FALSE, warning=FALSE}
Packages <- c( 'MASS', 'dplyr' ,'tidyverse', 'GGally', 'ISLR', 'caret',
'class','ROCR', 'boot', 'glmnet', 'pROC' ,'tree','randomForest', 'e1071',
'skimr')
lapply ( Packages , library, character.only  = TRUE)
```

```{r include=FALSE ,message=FALSE, warning=FALSE}
df=read_csv('/Users/angelsilvar/Desktop/STATS 473/Final Project/
            SweatShop.csv')
df = as_tibble(df)
```


```{r include=TRUE ,message=FALSE, warning=FALSE}
factors = c( "date", "quarter", "day", "department",
        "no_of_style_change", "team")

df1 = df %>%
  mutate_at( factors , factor) %>%
  dplyr::select(-c(department, date, targeted_productivity))%>% drop_na()
```

<!-- ```{r include=TRUE ,message=FALSE, warning=FALSE} -->
<!-- n = nrow(df1) -->
````

````
<!-- prop = .8 -->
<!-- set.seed(123) -->
<!-- train_id = sample(1:n, size = n*prop, replace = FALSE) -->
<!-- test_id  = (1:n)[-which(1:n %in% train_id)] -->
<!-- train_set = df[train_id,] -->
<!-- test_set  = df[ test_id,] -->
<!-- ``` -->

```{r include=TRUE ,message=FALSE, warning=FALSE}
# Set a threshold of 80% for productivity to create binary
    #response variable
logit_df = df1 %>%
  mutate (productivity = ifelse ( actual_productivity >=.8 ,1,0))%>%
  dplyr::select(-c(actual_productivity))%>% drop_na() %>%
  mutate_at('productivity',factor)
logit_df
```


```{r include=TRUE ,message=FALSE, warning=FALSE}
# Check for unique values of each predictor
#lapply(df1[,], unique)
k = nrow(logit_df)
prop = .8
set.seed(123)
train_id = sample(1:k, size = k*prop, replace = FALSE)
test_id  = (1:k)[-which(1:k %in% train_id)]
train_set = logit_df[train_id,]
test_set  = logit_df[ test_id,]
```


```{r include=TRUE ,message=FALSE, warning=FALSE}
logit = glm( formula = productivity ~. ,data = train_set,family ="binomial")
summary(logit)
```


```{r include=TRUE ,message=FALSE, warning=FALSE}
logit_best = glm( formula = productivity ~  team + over_time
                  + smv + incentive + no_of_style_change + no_of_workers ,
                  family = 'binomial' ,data = train_set)

summary(logit_best)
logit_pred = predict( logit_best , test_set , type = "response" )
tb_log = table(predict_status = logit_pred  > 0.5 , true_status =
            test_set$productivity == 1  )
(tb_log[1,1] + tb_log[2,2])/sum(tb_log)

```


```{r include=TRUE ,message=FALSE, warning=FALSE}
glm_pred2 = predict( logit_best ,test_set )
glm.pred2 = prediction(glm_pred2, test_set$productivity)
glm.perf = performance(glm.pred2, "tpr", "fpr")
````
````

````
plot(glm.perf, main = "ROC Curve")
abline(0, 1, lty=3)

auc = as.numeric(performance(glm.pred2, "auc")@y.values)
auc
```


### LDA Model

```{r include=TRUE ,message=FALSE, warning=FALSE}
lda_fit = lda(formula = productivity ~. , data = train_set )
lda_fit
```


```{r include=TRUE ,message=FALSE, warning=FALSE}

lda_pred_class = predict(lda_fit, test_set)$class
tb_lda = table(predict_status = lda_pred_class,
            true_status=test_set$productivity)
tb_lda

(tb_lda[1,1] + tb_lda[2,2])/sum(tb_lda)



lda_pred = predict(lda_fit, test_set)
lda_pred_post = lda_pred$posterior[,2]
pred = prediction(lda_pred_post, test_set$productivity)
perf = performance(pred, "tpr", "fpr")
plot(perf, main = "ROC Curve")
abline(0, 1, lty=3)

auc1 = as.numeric(performance(pred, "auc")@y.values)
auc1

```


### Lasso Model

```{r include=TRUE ,message=FALSE, warning=FALSE}
set.seed(123)
xmat = model.matrix(productivity  ~., data = train_set )[,-1]
xmat = apply(xmat, 2, function (x) scale(x, center=FALSE))

mod.lasso = glmnet(xmat, train_set$productivity ,family = "binomial"
                ,alpha=1)
plot(mod.lasso, xvar = "lambda", label = TRUE)


cv.out = cv.glmnet(xmat, train_set$productivity,
                family = "binomial", alpha = 1 , nfolds=10)

best.lambda = cv.out$lambda.min
````

```
best.lambda

pcoefs = predict(mod.lasso, s = best.lambda, type = "coefficients")
pcoefs
xmat2 = model.matrix(productivity  ~., data = test_set )[,-1]
xmat2 = apply(xmat2, 2, function (x) scale(x, center=FALSE))
lasso_pred = predict(mod.lasso,newx = xmat2, s=best.lambda ,
                type = "class" )
#######
tb_lasso = table(predict_status = lasso_pred == '1' , true_status =
            test_set$productivity == 1  )
(tb_lasso[1,1] + tb_lasso[2,2])/sum(tb_lasso)


```


Tree Models: Classification Tree

```{r  include=TRUE ,message=FALSE, warning=FALSE}
train_set_tree = train_set %>%
  mutate(class_productivity = ifelse(
                        productivity == 1 , "Yes", "No"))%>%
  dplyr::select(-productivity)

train_set_tree = train_set_tree %>%
  mutate_at('class_productivity' , factor)



test_set_tree = test_set %>%
  mutate(class_productivity = ifelse(productivity == 1 , "Yes", "No"))%>%
  dplyr::select(-productivity)

test_set_tree = test_set_tree %>%
  mutate_at('class_productivity', factor)

skim(train_set_tree)

mod.tree2 = tree( class_productivity  ~. , data = train_set_tree )

summary(mod.tree2)
mod.tree2
plot(mod.tree2)
text(mod.tree2, pretty = 0)
```



```{r  include=TRUE ,message=FALSE, warning=FALSE}
set.seed(123)
cv.out = cv.tree(mod.tree2)
prune.mod = prune.misclass(mod.tree2, best =
                    cv.out$size[which.min(cv.out$dev)])
summary(prune.mod)
prune.mod
```

```
plot(prune.mod)
text(prune.mod, pretty = 0)
```

```{r,message=FALSE,warning=FALSE}
#Confusion matrix - Full Tree
tree_pred_class = predict(mod.tree2, newdata = test_set_tree, type ="class")
tb_full = table(predict_status =
                    tree_pred_class,true_status=test_set_tree$class_productivity)
tb_full
(tb_full[1,1] + tb_full[2,2])/sum(tb_full)

#Confusion matrix - Pruned Tree
tree_pred_class = predict(prune.mod,newdata= test_set_tree, type ="class")
tb_prune = table(predict_status =
            tree_pred_class,true_status=test_set_tree$class_productivity)
tb_prune
(tb_prune[1,1] + tb_prune[2,2])/sum(tb_prune)

```


Tree Based Model: Bagging, Random Forest, Gradient Boosting

Random Forest
```{r  include=TRUE ,message=FALSE, warning=FALSE}

set.seed(123)
p = ncol(train_set) - 1

rf_fit = randomForest(productivity  ~ ., data = train_set,
                      mtry = round(sqrt(p)), importance = TRUE)
rf_fit
summary(rf_fit)
varImpPlot(rf_fit, main = "Variable Importance", type = 2 )

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf

(tb_rf[1,1] + tb_rf[2,2])/sum(tb_rf)

```
Incentive
```{r  include=TRUE ,message=FALSE, warning=FALSE}
avg  = data.frame( day = train_set$day[1]  ,
                   quarter = train_set$quarter[1],
                   no_of_style_change = train_set$no_of_style_change[1],
                   team = train_set$team[1]
                    , smv = 24.23, wip = 1190 ,
                   incentive = c(44,0 ,138, 17 , 71 ) ,
                over_time = 6508, idle_time = 1.2 ,
                   idle_men =0.63   , no_of_workers = 52.44 )
```

```
# 17 = AVG - ST.D
# 71 = AVG + ST.D


as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,true = test_set$productivity)
tb_rf


```
#Over time
```{r include=TRUE ,message=FALSE, warning=FALSE}

avg = data.frame( day = train_set$day[1] ,
                  quarter = train_set$quarter[1],
                  no_of_style_change = train_set$no_of_style_change[1],
                  team = train_set$team[1]
                  ,
                  smv = 24.23, wip = 1190 ,
                  incentive = 44 ,
                  over_time = c( 4567 ,0, 25920, 4567+ 2864, 4567 - 2864) ,
                  idle_time = 1.2 ,
                  idle_men =0.63  , no_of_workers = 52.44 )





as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf
```
idle_time
```{r include=TRUE ,message=FALSE, warning=FALSE}
avg = data.frame( day = train_set$day[1] ,
                  quarter = train_set$quarter[1],
                  no_of_style_change = train_set$no_of_style_change[1],
                  team = train_set$team[1]
                  , smv = 24.23, wip = 1190 ,
                  incentive = 44 , over_time = 6508,
                  idle_time = c( 1.2, 0 , 300, 0 , 1.2+16) ,
                  idle_men = 0.63
                  ,no_of_workers = 52.44 )
```

```
as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf
```


idle_men
```{r include=TRUE ,message=FALSE, warning=FALSE}
avg = data.frame( day = train_set$day[1] ,
                  quarter = train_set$quarter[1],
                  no_of_style_change = train_set$no_of_style_change[1],
                  team = train_set$team[1],
                  smv = 24.23, wip = 1190 ,
                  incentive = 44 , over_time = 6508,
                  idle_time = 1.2 ,
                  idle_men = c( 0.63, 0 , 45, 0 , 0.63 + 4.28)
                  ,no_of_workers = 52.44 )





as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf
```


idle_men & Idle_time
```{r include=TRUE ,message=FALSE, warning=FALSE}
avg = data.frame( day = train_set$day[1] , quarter = train_set$quarter[1],
               no_of_style_change =
                  train_set$no_of_style_change[1],
                  team = train_set$team[1],
                   smv = 24.23, wip = 1190 ,
                  incentive = 44 , over_time = 6508,
                  idle_time = c( 1.2, 0 , 300, 0 , 1.2+16) ,
                  idle_men = c( 0.63, 0 , 45, 0 , 0.63 + 4.28)
                  ,no_of_workers = 52.44 )
```

```
as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf
```


#Over time + Incentives
```{r  include=TRUE ,message=FALSE, warning=FALSE}

avg  = data.frame( day = train_set$day[1] ,
                   quarter = train_set$quarter[1],
                   no_of_style_change = train_set$no_of_style_change[1],
                   team = train_set$team[1]
                   , smv = 24.23, wip = 1190 ,
                   incentive = c (44,0 ,138, 17 , 71 )
                   , over_time = c( 4567 ,0, 25920, 4567+ 2864, 4567 - 2864)
                   , idle_time = 1.2 ,
                   idle_men =0.63  , no_of_workers = 52.44 )





as.tibble(avg)
yhat.test.rf = predict(rf_fit, avg, type = "class")
yhat.test.rf

yhat.test_rf = predict(rf_fit, test_set, type = "class")
tb_rf = table(pred = yhat.test_rf,
true = test_set$productivity)
tb_rf

```


Bagging Model

```{r  include=TRUE ,message=FALSE, warning=FALSE}

set.seed(123)
p = ncol(train_set) - 1

bag_fit = randomForest(productivity  ~ ., data = train_set,
                      mtry = p, importance = TRUE)
bag_fit
varImpPlot(bag_fit, main = "Variable Importance", type = 2 )
#plot(bag_fit)
```

```
```{r  include=TRUE ,message=FALSE, warning=FALSE}
yhat.test_bag = predict(bag_fit, test_set, type = "class")
tb_bag = table(predicted_status = yhat.test_bag,
               true_status = test_set$productivity)
tb_bag

(tb_bag[1,1] + tb_bag[2,2])/sum(tb_bag)

```

```{r  include=TRUE ,message=FALSE, warning=FALSE}

fhat_rf = predict(rf_fit, newdata = test_set, type = 'class' )
tb_rf = table(predict_status= fhat_rf,
              true_status = test_set$productivity)
tb_rf
(tb_rf[1,1] + tb_rf[2,2])/sum(tb_rf)

```
#Gradient Boosting

```{r  include=TRUE ,message=FALSE, warning=FALSE}
library(gbm)

set.seed(123)
boost_fit = gbm( productivity ~ ., train_set, n.trees = 100, shrinkage = 0.1,
                interaction.depth = 1,
                distribution = "bernoulli")

phat.test_boost = predict(boost_fit, test_set, type = "response")


yhat.test_boost = ifelse(phat.test_boost > 0.5, 1, 0)
tb_boost = table(pred = yhat.test_boost, true = test_set$productivity)

tb_boost

(tb_boost[1, 1] + tb_boost[2, 2])/sum(tb_boost)
```

```{r  include=TRUE ,message=FALSE, warning=FALSE}

set.seed(123)
grid = expand.grid(
n.trees_vec = c(100, 200),
shrinkage_vec = c(0.2, 0.1, 0.06, 0.05, 0.04, 0.02, 0.01),
interaction.depth_vec = c(1, 2, 3),
miss_classification_rate = NA,
time = NA
)
```

```{r  include=TRUE ,message=FALSE, warning=FALSE}
for(i in 1:nrow(grid)){ time = system.time({
```
```

```
boost_fit = gbm(productivity ~ ., train_set2, n.trees = grid$n.trees_vec[i],
              shrinkage = grid$shrinkage_vec[i],
              interaction.depth = grid$interaction.depth_vec[i], distribution =
              "bernoulli", cv.folds = 5)} )
              grid$miss_classification_rate[i]
              =boost_fit$cv.error[which.min(boost_fit$cv.error)]
              grid$time[i] = time[["elapsed"]]
}

grid %>% arrange(miss_classification_rate)

boost_fit_best = gbm(productivity ~ ., train_set2, n.trees = 200, shrinkage = 0.05,
                     = 3,distribution = "bernoulli")
phat.test_boost_best = predict(boost_fit_best, test_set2 , type = "response")
yhat.test_boost_best = ifelse(phat.test_boost_best > 0.5, 1, 0)
tb_boost_best = table(pred = yhat.test_boost_best,
true = test_set2$productivity)
tb_boost_best
sum(diag(tb_boost_best))/sum(tb_boost_best)
```


Support Vector Machine
```{r  include=TRUE ,message=FALSE, warning=FALSE}

set.seed(123)
tune_svm_linear = tune(svm, productivity ~., data = train_set, kernel = "linear",
                    ranges = list(cost = 10^ seq(-3, 2, length.out=6)))
summary(tune_svm_linear)

tune_svm_radial = tune(svm, productivity ~., data = train_set, kernel = "radial",
              gamma = 1, ranges = list(cost = 10^ seq(-3, 2, length.out=6)))
summary(tune_svm_radial)

tune_svm_poly = tune(svm, productivity ~., data = train_set,
                    kernel = "radial", degree = 3,
ranges = list(cost = 10^ seq(-3, 2, length.out=6)))
summary(tune_svm_poly)
```
```{r  include=TRUE ,message=FALSE, warning=FALSE}

svm_fit = svm(productivity ~., data = train_set, kernel = "linear",
                cost = .1, scale = FALSE)
svm_fit_radia = svm(productivity ~., data = train_set,
                kernel = "radial", cost = 10, scale = FALSE)
svm_fit_poly  = svm(productivity ~., data = train_set,
                kernel = "polynomial", cost = 10,scale = FALSE)

summary(svm_fit)
summary(svm_fit_poly)
summary(svm_fit_radia)

```

```
```

Confusion Matrix for Maximal Machine Classifier,
    ####Support Vector Classifier, Support Vector Machine
```{r  include=TRUE ,message=FALSE, warning=FALSE}

yhat_test = predict(svm_fit, test_set)
tb_svm = table(pred = yhat_test, truth = test_set$productivity)
tb_svm
(tb_svm[1,1] + tb_svm[2,2])/sum(tb_svm)

yhat_test = predict(svm_fit_radia, test_set)
tb_svm2 = table(pred = yhat_test, truth = test_set$productivity)
tb_svm2
(tb_svm2[1,1] + tb_svm2[2,2])/sum(tb_svm2)

yhat_test = predict(svm_fit_poly, test_set)
tb_svm3 = table(pred = yhat_test, truth = test_set$productivity)
tb_svm3
(tb_svm3[1,1] + tb_svm3[2,2])/sum(tb_svm3)




```
```
```