

Machine Predictive Maintenance Classification

ST406 FINAL PROJECT

S/18/813
Tharindu Kariyawasam

1. Introduction

Predictive maintenance (PdM) leveraging statistical learning, IoT, and artificial intelligence techniques has gained significant attention in recent years, particularly within the context of Industry 4.0. The application of PdM involves utilizing sensor data and advanced algorithms to predict equipment failures before they occur, thereby minimizing downtime and maintenance costs. Major purpose of this study is to build a classification model to accurately predict machinery failures in an industrial setting.

1.1. Major Question

- How can statistical learning models be effectively utilized to predict machinery failures?

1.2. Objectives

- Develop a classification model to accurately predict machinery failures.
- Evaluate Model Performance.
- Interpret the model's predictions, and explain feature importance and their impact on failure predictions.

1.3. Literature Review

Researchers have carried out many studies in this field. For instance, Kane, Kore, Khandale, Nigade, and Joshi (2022). propose a web application that uses machine sensor data to forecast potential machine downtimes through neural network models. Similarly, Paolanti et al. (2018). describe a machine learning approach for PdM on a cutting machine, employing Random Forest algorithms within the Azure machine learning studio to create dynamic maintenance management rules. In another study, Susto, Schirru, Pampuri, McLoone, and Beghi (2015). introduce a

multiple-classifier system aimed at improving maintenance management in semiconductor manufacturing by minimizing operational costs. This approach demonstrates the effectiveness of integrating various classifiers for fault prediction. Kanawaday and Sane (2017) explore the ARIMA forecasting model on time series data from sensors on a slitting machine, highlighting its potential in predicting failures and enhancing manufacturing processes.

Pagano (2023) work presents a novel PdM model combining Long Short-Term Memory (LSTM) neural networks with Bayesian inference, emphasizing its application in industrial plant maintenance. Collectively, these studies underscore the diverse methodologies and significant advancements in statistical & machine learning applications for predictive maintenance.

2. Methodology

2.1. Description of Dataset

The dataset consists of 10 000 data points stored as rows with 14 features in columns

- UID: unique identifier ranging from 1 to 10000
- productID: consisting of a letter L, M, or H for low (50% of all products), medium (30%), and high (20%) as product quality variants and a variant-specific serial number
- air temperature [K]: generated using a random walk process later normalized to a standard deviation of 2 K around 300 K
- process temperature [K]: generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.
- rotational speed [rpm]: calculated from power of 2860 W, overlaid with a normally distributed noise
- torque [Nm]: torque values are normally distributed around 40 Nm with an $\bar{f} = 10$ Nm and no negative values.
- tool wear [min]: The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

- Target: Failure or Not
- Failure Type: Type of Failure

2.2. Statistical Techniques & Assumptions

2.2.1. Naive Bayes

A probabilistic classifier based on Bayes' Theorem, it calculates the probability of each class given the features and assigns the class with the highest posterior probability.

The probability of a class C given features X_1, X_2, \dots, X_n is calculated using Bayes' Theorem

$$P(C | X_1, X_2, \dots, X_n) = \frac{P(C) \cdot P(X_1 | C) \cdot P(X_2 | C) \cdots P(X_n | C)}{P(X_1, X_2, \dots, X_n)}$$

Assumptions:

- Conditional independence of each feature given the class label.
- Continuous features follow Gaussian distribution.

2.2.2. Logistic Regression

Statistical method used to model binary outcomes. It predicts the probability of an event occurring based on one or more variables

$$P(Y = 1 | X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}}$$

Assumptions:

- There is a linear relationship between the log-odds of the outcome and predictor variables.
- Observations are independent of each other.
- Predictors should not be highly correlated with each other.
- The variance of error terms is constant across all levels of the independent variables.

2.2.3. Bayesian Model Selection

Evaluates different models based on their posterior probabilities. It balances model fit with model complexity by penalizing larger models.

Assumptions:

- Assumes uniform prior over the model space.
- Residuals are assumed to follow a normal distribution.
- Assumes the predictors have a linear relationship with the outcome variable.
- Observations are independent of each other.

2.2.4. Confusion Matrix and Model Evaluation

Confusion matrix is used to evaluate the performance of the classification models. Can derive metrics such as accuracy, precision, recall and F1-score from the confusion matrix.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives
- Accuracy gives an overall effectiveness of the model.
- Precision focuses on the quality of positive predictions.
- Recall focuses on the ability of the model to capture all positive instances.
- F1-Score balances precision and recall, particularly useful when the class distribution is imbalanced.

3. Results & Discussion

3.1. Distributions of Numerical Features

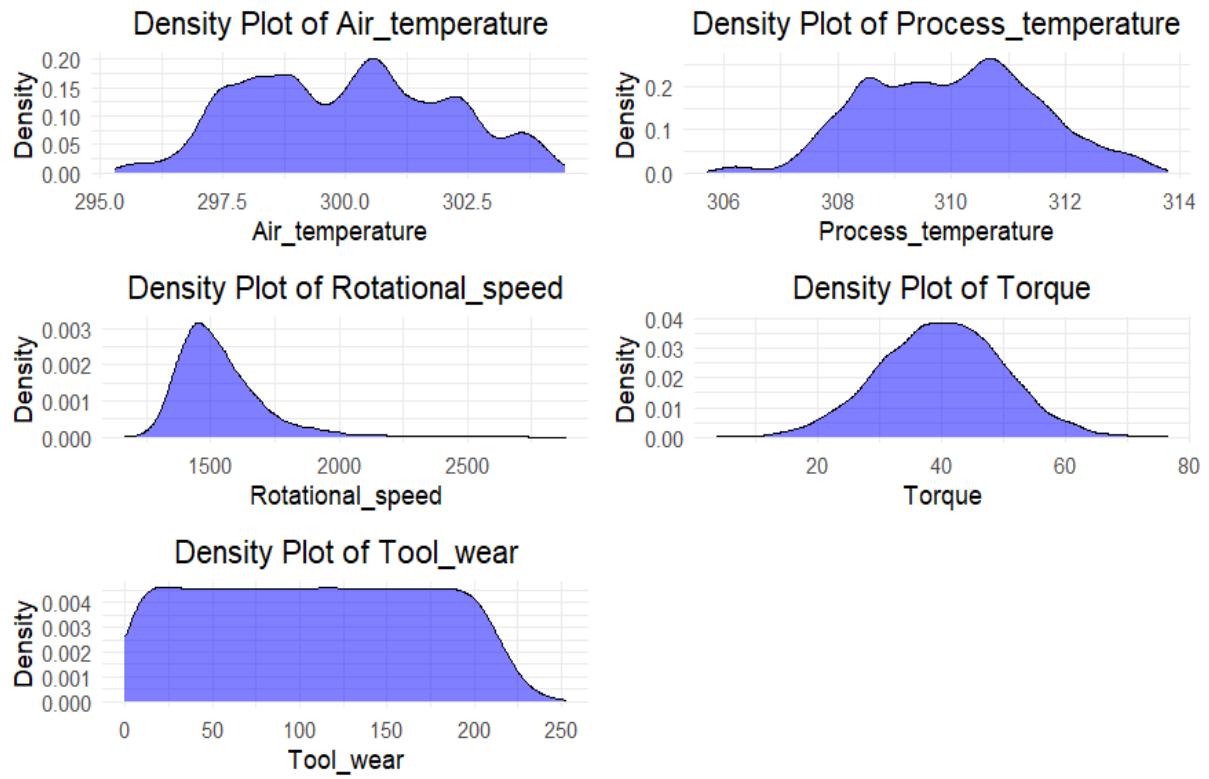


Figure 1 : Distribution Plots of Numerical Variables

3.2. Distributions of Categorical Features

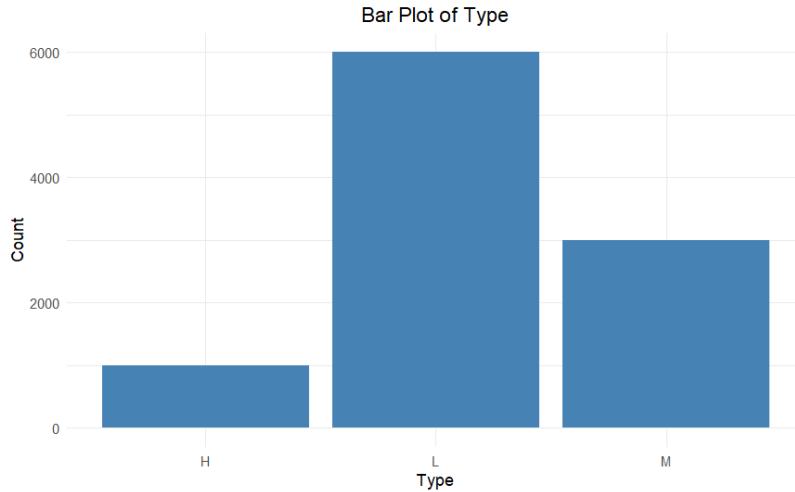


Figure 2 : Bar Graph of Machine Type

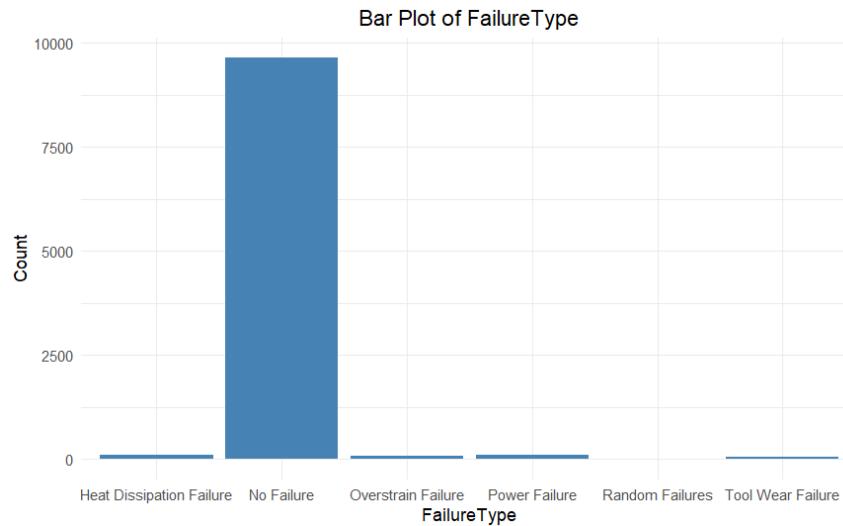


Figure 3 : Bar Graph of Failure Types

3.3. Distributions of Target Variable

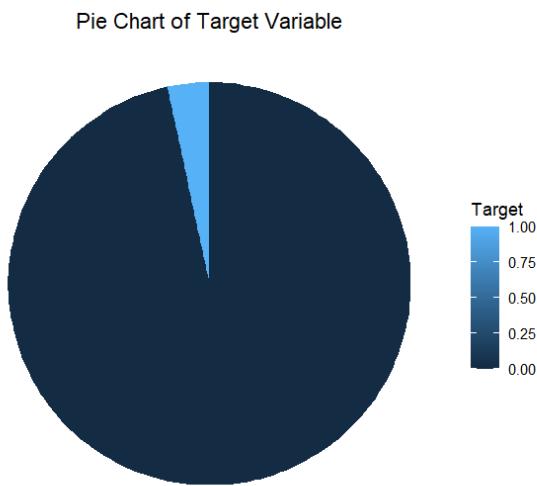


Figure 4 : Pie Chart For Target Variable

3.4. Correlation Between Feature Variables

Table 1 : Correlation of Feature Variables

	Type	Air_temperature	Process_temperature	Rotational_speed	Torque	Tool_wear
Type	1.0000	0.0176	0.0134	-0.0027	0.0040	-0.0033
Air_temperature	0.0176	1.0000	0.8761	0.0227	-0.0138	0.0139
Process_temperature	0.0134	0.8761	1.0000	0.0193	-0.0141	0.0135
Rotational_speed	-0.0027	0.0227	0.0193	1.0000	-0.8750	0.0002
Torque	0.0040	-0.0138	-0.0141	-0.8750	1.0000	-0.0031
Tool_wear	-0.0033	0.0139	0.0135	0.0002	-0.0031	1.0000

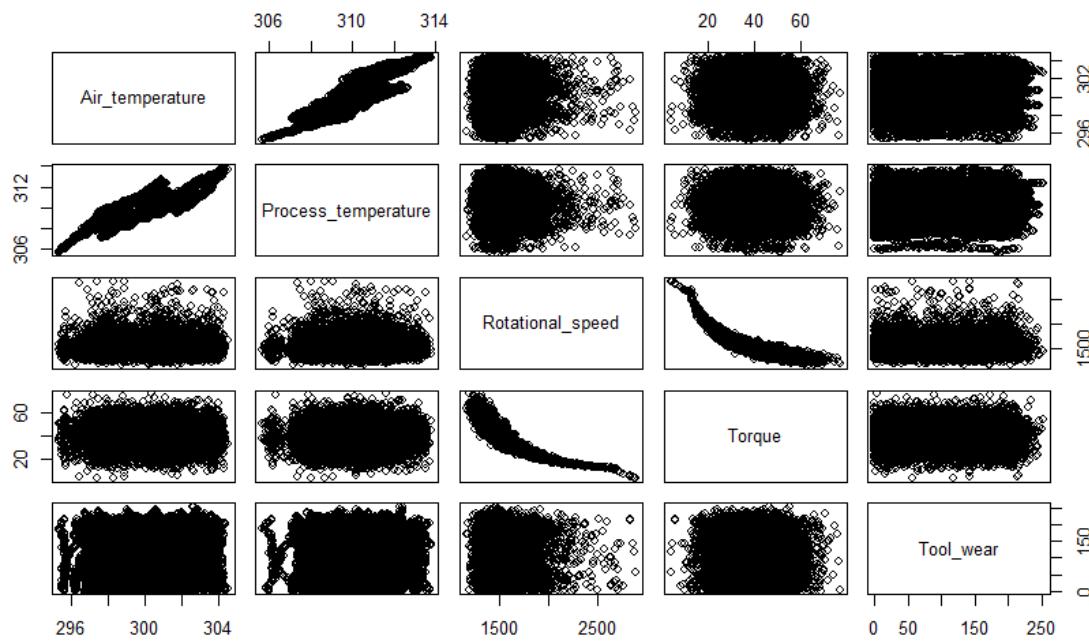


Figure 5 : Correlation Plot for Feature Variables

4. Conclusion & Recommendation

4.1. Naive Bayes Model

```
Target ~ Type + Air_temperature + Process_temperature + Rotational_speed + Torque + Tool_wear

Accuracy : 0.956
95% CI  : (0.9414, 0.9679)
No Information Rate : 0.966
P-Value [Acc > NIR] : 0.96223

Kappa : 0.1641

McNemar's Test P-Value : 0.05002

Sensitivity : 0.9845
Specificity : 0.1471
Pos Pred Value : 0.9704
Neg Pred Value : 0.2500
Prevalence : 0.9660
Detection Rate : 0.9510
Detection Prevalence : 0.9800
Balanced Accuracy : 0.5658
```

The model has high accuracy (95.6%) which indicates predictions made by the model are correct 95.6% of the time. But as the Kappa value is low (0.1641) it indicates that model is not capturing the less frequent class well. Also sensitivity (0.9845) and specificity (0.1471) also give the same conclusion that the model performs well for the majority class but performs poorly for the minority class. McNemar's Test P-Value is 0.05002 this indicate that there might be statistically significant difference in the model's error rates between the two classes. Due to the imbalance of the target variable we get lower balanced accuracy (0.5658) than the overall accuracy.

To overcome this imbalance issue we can try several methods,

- Resampling Techniques like oversampling the minority class and under sampling the majority class.
- Introduce class weights
- Use ensemble methods like XGBoost which naturally handle imbalanced data.

4.2. Logistic Regression Model

```
Accuracy : 0.971
95% CI : (0.9586, 0.9805)
No Information Rate : 0.966
P-Value [Acc > NIR] : 0.2192

Kappa : 0.2499

McNemar's Test P-Value : 1.999e-07

Sensitivity : 1.0000
Specificity : 0.1471
Pos Pred Value : 0.9709
Neg Pred Value : 1.0000
Prevalence : 0.9660
Detection Rate : 0.9660
Detection Prevalence : 0.9950
Balanced Accuracy : 0.5735
```

Logistic regression model also has the same performance as Naïve Bayes model.

4.3. Backward Stepwise Regression with BIC

```
Start: AIC=1789.24
Target ~ Air_temperature + Process_temperature + Rotational_speed +
Torque + Tool_wear

Df Deviance    AIC
<none>          1734.6 1789.2
- Process_temperature  1   1793.1 1838.6
- Air_temperature      1   1850.4 1895.9
- Tool_wear             1   1877.3 1922.8
- Rotational_speed      1   2114.2 2159.7
- Torque                 1   2437.2 2482.7

Call:
glm(formula = Target ~ Air_temperature + Process_temperature +
Rotational_speed + Torque + Tool_wear, family = binomial,
data = train_set)

Deviance Residuals:
Min     1Q     Median     3Q     Max
-2.0511 -0.1994 -0.1071 -0.0597  3.6420

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.86229  0.12938 -37.581 < 2e-16 ***
Air_temperature 1.54150  0.15236 10.117 < 2e-16 ***
Process_temperature -1.11040 0.15127 -7.341 2.12e-13 ***
Rotational_speed   2.06268  0.09875 20.887 < 2e-16 ***
Torque            2.77474  0.11905 23.307 < 2e-16 ***
Tool_wear          0.83962  0.07575 11.085 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2664.2 on 8999 degrees of freedom
Residual deviance: 1734.6 on 8994 degrees of freedom
AIC: 1746.6

Number of Fisher Scoring iterations: 8
```

As Residual Deviance is less than the Null Deviance it indicate that model explains a significant portion of variance in the data. Each coefficient shows the log-odds of the dependent variable associated with unit increase in the respective predictor. Since P-values of all predictors are < 0.05 they are

strongly related to target variable. And the Fisher Scoring iterations: 8 indicate algorithm successfully found the optimal solution after 8 iterations.

4.4. Bayesian Model Selection Using BIC

```

Call:
bas.lm(formula = Target ~ Type + Air_temperature + Process_temperature +
       Rotational_speed + Torque + Tool_wear, data = train_set,
       prior = "BIC", modelprior = uniform())

Marginal Posterior Inclusion Probabilities:
Intercept              Type      Air_temperature Process_temperature   Rotational_speed        Torque        Tool_wear
1.000000          0.01318      1.000000          1.000000          1.000000          1.000000          1.000000

Call:
bas.lm(formula = Target ~ Type + Air_temperature + Process_temperature +
       Rotational_speed + Torque + Tool_wear, data = train_set,
       n.models = 1, prior = "BIC", modelprior = uniform())

Marginal Posterior Inclusion Probabilities:
Intercept              Type      Air_temperature Process_temperature   Rotational_speed        Torque        Tool_wear
1                      0                      0                      0                      0                      0                      0
                                         Post Mean    Post SD    CI Lower    CI Upper
Intercept          0.03388889 0.001907414 0.03014992 0.03762786
Type               0.00000000 0.000000000 0.00000000 0.00000000
Air_temperature    0.00000000 0.000000000 0.00000000 0.00000000
Process_temperature 0.00000000 0.000000000 0.00000000 0.00000000
Rotational_speed   0.00000000 0.000000000 0.00000000 0.00000000
Torque             0.00000000 0.000000000 0.00000000 0.00000000
Tool_wear          0.00000000 0.000000000 0.00000000 0.00000000

```

This approach selects the simpler models by penalizing models with more parameters. The first set of outputs shows the marginal posterior inclusion probabilities. Here predictor variable Type has a value of 0.01318. This indicates that variable Type is unlikely to be an important predictor in the model. All other predictor variables and intercept has a value of 1 indicating that they are highly likely to be included in the model. According to the BIC criterion, the simplest model is the best model, therefore second output has 0's for all predictor variables and 1 for intercept. Suggesting that none of the predictors significantly contribute to predicting the outcome under the BIC criterion.

5. References

Kanawady, A., & Sane, A. (2017, November). Machine learning for predictive maintenance of industrial machines using iot sensor data. In 2017 8th ieee international conference on software engineering and service science (icsess) (pp. 87–90).

doi:10.1109/ICSESS.2017.8342870

Kane, A., Kore, A., Khandale, A., Nigade, S., & Joshi, P. P. (2022). Predictive maintenance using machine learning.

Pagano, D. (2023). A predictive maintenance model using long short-term memory neural networks and bayesian inference. Decision Analytics Journal.

Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., & Loncarski, J. (2018). Machine learning approach for predictive maintenance in industry 4.0. In 2018 ieee international conference on mechatronic and embedded systems and applications (mesa). doi:10.1109/MESA.2018.8449150

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifiers approach. IEEE Transactions on Industrial Informatics.

6. Appendices

6.1. Dataset

Table 2 : Data set

UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
6	M14865	M	298.1	308.6	1425	41.9	11	0	No Failure
7	L47186	L	298.1	308.6	1558	42.4	14	0	No Failure
8	L47187	L	298.1	308.6	1527	40.2	16	0	No Failure
9	M14868	M	298.3	308.7	1667	28.6	18	0	No Failure
10	M14869	M	298.5	309.0	1741	28.0	21	0	No Failure

6.2. Code

```
library(readr)
library(ggplot2)
library(ggally)
library(car)
library(bas)
library(vgam)
library(catools)
library(rstanarm)
library(e1071)
library(caret)

data <- read_csv("../data/predictive_maintenance.csv")

str(data)
summary(data)

sum(is.na(data))

colnames(data)
colnames(data) <- c("udi", "productid", "type", "air_temperature",
"process_temperature", "rotational_speed", "torque", "tool_wear", "target",
"failuretype")

data$type <- as.factor(data$type)
levels(data$type)
data$type<-as.numeric(data$type)

# set a seed
set.seed(123)

# split the data
split <- sample.split(data$target, splitratio = 0.9)
train_set <- subset(data, split == true)
test_set <- subset(data, split == false)

nrow(train_set)
nrow(test_set)

# fit the naive bayes model with binary outcome
model_nb <- naivebayes(
  target ~
  type+air_temperature+process_temperature+rotational_speed+torque+tool_wear,
  data = train_set
)

# make predictions on the test set
predictions <- predict(model_nb, newdata = test_set)
```

```

# convert target in the test set to a factor
test_set$target <- as.factor(test_set$target)

# check if both have the same levels
levels(predictions) <- levels(test_set$target)

# evaluate the model using a confusion matrix
confusionmatrix(predictions, test_set$target)

# logistic regression model
model_logit <- glm(target ~ air_temperature + process_temperature +
rotational_speed + torque + tool_wear,
                     data = train_set, family = binomial)

# make predictions on the test set
predictions_logit <- predict(model_logit, newdata = test_set, type =
"response")

# convert probabilities to class labels
predicted_class <- ifelse(predictions_logit > 0.5, "1", "0") # assuming
"1" = failure, "0" = no failure

# convert to factor to match test set levels
predicted_class <- factor(predicted_class, levels =
levels(test_set$target))

# evaluate the model
confusionmatrix(predicted_class, test_set$target)

# backward stepwise regression using bic
model1 <- step(model_logit, direction = "backward", k =
log(nrow(train_set)), trace = 1)

# summary of the best model after backward elimination
summary(model1)

# bayesian model selection using bic prior
pred.bic <- bas.lm(target ~ type + air_temperature + process_temperature +
rotational_speed + torque + tool_wear,
                     data = train_set, prior = "bic", modelprior = uniform())
print(pred.bic)

# get the best model based on bic
pred.bestbic <- bas.lm(target ~ type + air_temperature +
process_temperature + rotational_speed + torque + tool_wear,
                     data = train_set, prior = "bic", modelprior =
uniform(), n.models = 1)
print(pred.bestbic)

# extract posterior coefficients from the best bic model

```

```
pred.coef <- coef(pred.bestbic)

# get confidence intervals for the coefficients
out <- confint(pred.coef)[,1:2]

# combine posterior means, standard deviations, and confidence intervals
coef.bic <- cbind(pred.coef$postmean, pred.coef$postsd, out)
colnames(coef.bic) <- c("post mean", "post sd", "ci lower", "ci upper")

# view the posterior coefficient results
print(coef.bic)
```