

Rival Fitrah Dermawan | Hotniel Diasi Sianturi | Anwar Faiz Fauzi

2025-12-12

Daftar Isi

1 Analisis Segmentasi dan Prediksi Churn Menggunakan Dataset Spotify Customer Churn dengan Algoritma Random Forest	2
1.1 Latar Belakang Masalah	2
1.2 Rumusan masalah	2
1.3 Tujuan	2
2 Persiapan Environment	3
3 Load data dari csv	3
3.1 melakukan cek kualitas data	4
4 Data Preparation	6
4.1 mengganti data yang kotor	6
4.2 menghapus kolom yang tidak diperlukan	7
4.3 proses encoding untuk variabel numerik	8
5 Modeling	8
5.1 membagi data latih 70 dan data uji 30	8
5.2 Menangani Ketidakseimbangan Kelas dengan SMOTE	9
5.3 Membangun model random forest	9
5.4 visualisasi data sebelum dan sesudah menggunakan SMOTE	10
6 Evaluation	11
7 Kesimpulan dan saran	13
7.1 Kesimpulan	13
7.2 saran	14

Daftar Gambar

**Analisis Segmentasi dan Prediksi Churn Menggunakan Dataset
Spotify Customer Churn
dengan Algoritma Random Forest**

1 Analisis Segmentasi dan Prediksi Churn Menggunakan Dataset Spotify Customer Churn dengan Algoritma Random Forest

1.1 Latar Belakang Masalah

Spotify merupakan salah satu platform streaming musik terbesar di dunia, dengan ratusan juta pengguna aktif yang terdiri dari pengguna gratis dan pengguna berlangganan Premium. Sebagai layanan digital, mempertahankan pelanggan menjadi hal yang sangat penting karena pengguna dapat dengan mudah berpindah ke platform lain. Kondisi ini membuat masalah customer churn—atau berhentinya pelanggan menggunakan layanan—menjadi perhatian utama bagi perusahaan.

Churn dapat berdampak langsung pada pendapatan dan keberlanjutan bisnis. Oleh sebab itu, dibutuhkan metode yang mampu memprediksi pengguna mana yang berisiko churn sehingga perusahaan dapat melakukan tindakan preventif. Dalam penelitian ini, dilakukan analisis dan prediksi churn menggunakan algoritma Random Forest pada dataset Spotify Customer Churn. Tujuan dari proses ini adalah mengidentifikasi pola perilaku pengguna serta faktor-faktor yang memengaruhi keputusan mereka untuk berhenti menggunakan layanan, sehingga dapat menjadi dasar dalam strategi retensi pelanggan.

1.2 Rumusan masalah

1. Bagaimana kondisi kualitas data pada dataset Spotify Customer Churn sebelum dilakukan proses analisis?
2. Faktor-faktor apa saja yang berpotensi memengaruhi seorang pengguna melakukan churn?
3. Bagaimana cara membangun model prediksi churn menggunakan algoritma Random Forest?
4. Seberapa baik performa model Random Forest dalam memprediksi churn pada pengguna Spotify?
5. Bagaimana hasil evaluasi model dapat digunakan sebagai dasar rekomendasi strategi retensi pengguna?

1.3 Tujuan

1. Menganalisis dan membersihkan dataset Spotify Customer Churn agar siap digunakan untuk proses pemodelan.
2. Mengidentifikasi variabel-variabel yang berpengaruh terhadap perilaku churn pengguna.
3. Membangun model prediksi churn menggunakan algoritma Random Forest.
4. Mengevaluasi performa model dengan menggunakan metrik seperti accuracy, precision, recall, dan F1-score.
5. Memberikan insight dan rekomendasi berdasarkan hasil analisis untuk membantu perusahaan dalam mengurangi tingkat churn.

2 Persiapan Environment

Langkah pertama yang dilakukan pada analisa ini adalah memanggil library yang diperlukan setelah melakukan instalasi pada library yang akan digunakan

```
# load library
library(tidyverse)
library(randomForest)
library(caret)
library(smotefamily)
library(ggplot2)
```

1. library tidyverse = library ini digunakan untuk memanipulasi dataset
2. library randomforest = library ini digunakan untuk membangun model random forest
3. libarry caret = library ini digunakan untuk membantu melakukan preprocessing data
4. library smotefamily = library ini digunakan untuk mengatasi ketidakseimbangan kelas pada variabel target "churned" menggunakan algoritma SMOTE (Synthetic Minority Over-sampling Technique)
5. library ggplot = digunakan untuk melakukan visualisasi data

3 Load data dari csv

```
# load dataset
spotify <- read_csv("data spotify_churn_dataset.csv", show_col_types = FALSE)

cat("6 Baris Data Pertama:\n")

## 6 Baris Data Pertama:

print(head(spotify))

## # A tibble: 6 x 10
##   user_id subscription_type country avg_daily_minutes number_of_playlists
##   <chr>    <chr>          <chr>           <dbl>                  <dbl>
## 1 user_1  Premium         US            135.                  4
## 2 user_2  Premium         PK            166.                  5
## 3 user_3  Free            DE            45.9                 3
## 4 user_4  Premium         PK            106                  0
## 5 user_5  Premium         US            89.6                 5
## 6 user_6  Free            IN            17.6                 0
## # i 5 more variables: top_genre <chr>, skips_per_day <dbl>,
## # support_tickets <dbl>, days_since_last_login <dbl>, churned <dbl>
```

```

cat("\nStruktur Dataset Awal:\n")

## 
## Struktur Dataset Awal:

glimpse(spotify)

## Rows: 1,000
## Columns: 10
## $ user_id           <chr> "user_1", "user_2", "user_3", "user_4", "user_5"~
## $ subscription_type <chr> "Premium", "Premium", "Free", "Premium", "Premiu~
## $ country            <chr> "US", "PK", "DE", "PK", "US", "IN", "UK", "BR", ~
## $ avg_daily_minutes <dbl> 134.9, 165.7, 45.9, 106.0, 89.6, 17.6, 43.7, 131~
## $ number_of_playlists <dbl> 4, 5, 3, 0, 5, 0, 2, 3, 5, 1, 1, 2, 1, 0, 4, 3, ~
## $ top_genre           <chr> "Electronic", "Pop", "Classical", "Jazz", "Count~
## $ skips_per_day       <dbl> 6, 8, 3, 7, 2, 6, 1, 7, 4, 6, 1, 7, 0, 2, 2, 7, ~
## $ support_tickets      <dbl> 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ days_since_last_login <dbl> 1, 12, 3, 3, 6, 7, 0, 1, 0, 10, 34, 25, 17, 7, 0~
## $ churned              <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, ~

```

pada bagian ini isi dari dataset ditampilkan sebanyak 6 baris pertama untuk melihat gambaran awal dari struktur isi dataset serta melihat format dari dataset, kemudian menggunakan fungsi `glimpse` untuk melihat struktir data lebih lengkap dab mendetail

3.1 melakukan cek kualitas data

```

# 1. Cek Data Hilang
cat("\nCek Data Hilang (NA):\n")

## 
## Cek Data Hilang (NA):

print(colSums(is.na(spotify)))

##          user_id    subscription_type        country
##                 0                  0                  0
## avg_daily_minutes number_of_playlists top_genre
##                 0                  0                  0
## skips_per_day     support_tickets  days_since_last_login
##                 0                  0                  0
##          churned
##                 0

```

```
# 2. Cek Data Duplikasi
cat("\nCek Duplikasi:\n")

## 
## Cek Duplikasi:

print(sum(duplicated(spotify)))

## [1] 0

# 3. Cek Struktur Data
cat("\nStruktur Dataset:\n")

## 
## Struktur Dataset:

str(spotify)

## #> #> spc_tbl_ [1,000 x 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## #>   $ user_id          : chr [1:1000] "user_1" "user_2" "user_3" "user_4" ...
## #>   $ subscription_type : chr [1:1000] "Premium" "Premium" "Free" "Premium" ...
## #>   $ country           : chr [1:1000] "US" "PK" "DE" "PK" ...
## #>   $ avg_daily_minutes : num [1:1000] 134.9 165.7 45.9 106 89.6 ...
## #>   $ number_of_playlists: num [1:1000] 4 5 3 0 5 0 2 3 5 1 ...
## #>   $ top_genre          : chr [1:1000] "Electronic" "Pop" "Classical" "Jazz" ...
## #>   $ skips_per_day      : num [1:1000] 6 8 3 7 2 6 1 7 4 6 ...
## #>   $ support_tickets     : num [1:1000] 0 0 0 0 1 2 1 0 0 0 ...
## #>   $ days_since_last_login: num [1:1000] 1 12 3 3 6 7 0 1 0 10 ...
## #>   $ churned            : num [1:1000] 0 0 0 0 0 1 0 0 0 0 ...
## #> - attr(*, "spec")=
## #>   .. cols(
## #>     ..   user_id = col_character(),
## #>     ..   subscription_type = col_character(),
## #>     ..   country = col_character(),
## #>     ..   avg_daily_minutes = col_double(),
## #>     ..   number_of_playlists = col_double(),
## #>     ..   top_genre = col_character(),
## #>     ..   skips_per_day = col_double(),
## #>     ..   support_tickets = col_double(),
## #>     ..   days_since_last_login = col_double(),
## #>     ..   churned = col_double()
## #>     .. )
## #> - attr(*, "problems")=<externalptr>
```

```

# 4. Cek Nilai Tidak Logis (Negatif) pada variabel numeric
cat("\nCek nilai negatif pada avg_daily_minutes:\n")

## 
## Cek nilai negatif pada avg_daily_minutes:

spotify %>% filter(avg_daily_minutes < 0) %>% print()

## # A tibble: 7 x 10
##   user_id subscription_type country avg_daily_minutes number_of_playlists
##   <chr>     <chr>          <chr>           <dbl>                  <dbl>
## 1 user_71  Free            PK             -37.2                  0
## 2 user_258 Free            IN             -12.7                  4
## 3 user_355 Free            CA             -16.6                  3
## 4 user_418 Free            IN             -27.6                  1
## 5 user_546 Free            UK             -8.9                   0
## 6 user_671 Free            CA             -19.1                  2
## 7 user_684 Free            FR             -6.2                   4
## # i 5 more variables: top_genre <chr>, skips_per_day <dbl>,
## #   support_tickets <dbl>, days_since_last_login <dbl>, churned <dbl>

cat("\nJumlah nilai negatif pada avg_daily_minutes:\n")

## 
## Jumlah nilai negatif pada avg_daily_minutes:

print(sum(spotify$avg_daily_minutes < 0, na.rm = TRUE))

## [1] 7

```

sebelum melakukan proses data preparation, kualitas data dilakukan pengecekan terlebih dahulu untuk melihat apakah terdapat data yang hilang, atau terdapat duplikasi data, kemudian melakukan pengecekan apakah terdapat nilai yang tidak logis, pada kasus ini ditemukan nilai tidak logis dimana terdapat nilai minus pada kolom **avg_daily_minutes**.

4 Data Preparation

4.1 mengganti data yang kotor

```

# cek data kotor avg_daily_minutes
spotify %>%
  filter(avg_daily_minutes < 0) %>%
  head()

```

```

## # A tibble: 6 x 10
##   user_id  subscription_type country avg_daily_minutes number_of_playlists
##   <chr>     <chr>          <chr>           <dbl>                  <dbl>
## 1 user_71   Free            PK              -37.2                  0
## 2 user_258  Free            IN              -12.7                  4
## 3 user_355  Free            CA              -16.6                  3
## 4 user_418  Free            IN              -27.6                  1
## 5 user_546  Free            UK              -8.9                   0
## 6 user_671  Free            CA              -19.1                  2
## # i 5 more variables: top_genre <chr>, skips_per_day <dbl>,
## #   support_tickets <dbl>, days_since_last_login <dbl>, churned <dbl>

sum(spotify$avg_daily_minutes < 0, na.rm = TRUE)

## [1] 7

# Hitung median dari nilai yang valid (>= 0)
median_val <- median(spotify$avg_daily_minutes[spotify$avg_daily_minutes >= 0], na.rm = TRUE)

# Ganti nilai negatif dengan median
spotify <- spotify %>%
  mutate(avg_daily_minutes = ifelse(avg_daily_minutes < 0, median_val, avg_daily_minutes))

cat("\nJumlah nilai negatif setelah perbaikan:\n")

## 
## Jumlah nilai negatif setelah perbaikan:

print(sum(spotify$avg_daily_minutes < 0))

```

[1] 0

Pada tahap data preparation, dilakukan pemeriksaan terhadap nilai-nilai yang tidak valid pada kolom **avg_daily_minutes**, Kolom ini merepresentasikan rata-rata durasi pemutaran musik harian pengguna, sehingga tidak mungkin nilai ini memiliki nilai negatif

Langkah yang dilakukan

- identifikasi nilai tidak valid menggunakan fungsi **filter()**
- mengganti nilai dengan menggunakan median agar menjaga konsistensi dari distribusi data. dengan menggunakan fungsi dari **mutate()** dan **ifelse()**
- Pengecekan ulang diakukan untuk memastikan data bersih dan tidak ada nilai yang tidak normal

4.2 menghapus kolom yang tidak diperlukan

```
# menghapus kolom user_id
spotify_clean <- spotify %>%
  select(-user_id)
```

pada bagian ini dilakukan kolom user_id tidak digunakan untuk proses pemodelan

4.3 proses encoding untuk variabel numerik

```
# Membuat model dummy/encoding
dummies_model <- dummyVars(churned ~ ., data = spotify_clean)

# Ubah dataset menjadi numeric semua
data_numeric <- predict(dummies_model, newdata = spotify_clean) %>% as.data.frame()

# atasi error pada penulisan di dataset
colnames(data_numeric) <- make.names(colnames(data_numeric))

# Mengembalikan variabel target 'churned' ke dataset
data_numeric$churned <- spotify_clean$churned
```

pada tahapan ini dilakukan proses variabel kategorikal menjadi numerik dikarenakan terdapat imbalance data sehingga akan dilakukan metode smote, sedangkan untuk melakukan smote ini diperlukan data dalam bentuk numerik

5 Modeling

5.1 membagi data latih 70 dan data uji 30

Pada tahapan ini dilakukan proses pembangian data latih sebanyak 70 dan data uji sebanyak 30

```
set.seed(123)

# data 70/30
train_index <- createDataPartition(data_numeric$churned, p = 0.7, list = FALSE)

train_data <- data_numeric[train_index, ]
test_data <- data_numeric[-train_index, ]

cat("\nJumlah kelas sebelum SMOTE:\n")

##
```

Jumlah kelas sebelum SMOTE:

```
print(table(train_data$churned))
```

```
##  
##    0    1  
## 568 132
```

5.2 Menangani Ketidakseimbangan Kelas dengan SMOTE

```
train_x <- train_data %>% select(-churned)  
train_y <- train_data$churned  
  
# SMOTE untuk menyeimbangkan jumlah kelas  
smote_output <- SMOTE(X = train_x, target = train_y, K = 5, dup_size = 0)  
  
train_smote <- smote_output$data  
  
# Rapikan kolom target  
colnames(train_smote)[ncol(train_smote)] <- "churned"  
colnames(train_smote) <- make.names(colnames(train_smote))  
  
# Pastikan target bertipe factor  
train_smote$churned <- as.factor(train_smote$churned)  
test_data$churned <- as.factor(test_data$churned)  
  
cat("\nJumlah kelas setelah SMOTE:\n")  
  
##  
## Jumlah kelas setelah SMOTE:  
  
print(table(train_smote$churned))
```

```
##  
##    0    1  
## 568 528
```

Untuk mengatasi data imbalance, digunakan metode SMOTE (Synthetic Minority Oversampling Technique) dari library smotefamily. Metode ini menghasilkan sampel sintetis (bukan duplikasi), sehingga kelas minoritas meningkat tanpa menyebabkan overfitting.

5.3 Membangun model random forest

```

rf_model <- randomForest(
  churned ~ .,
  data = train_smote,
  ntree = 500,
  mtry = 3,
  importance = TRUE
)

cat("\nHasil Model Random Forest:\n")

##  

## Hasil Model Random Forest:  

print(rf_model)  

##  

## Call:  

##   randomForest(formula = churned ~ ., data = train_smote, ntree = 500,           mtry = 3, importa  

##                 Type of random forest: classification  

##                           Number of trees: 500  

## No. of variables tried at each split: 3  

##  

##          OOB estimate of  error rate: 10.22%  

## Confusion matrix:  

##   0   1 class.error  

## 0 541  27  0.04753521  

## 1  85 443  0.16098485

```

pada tahapan ini dilakukan permodelan menggunakan random forest, random forest digunakan model ini karena cocok untuk dataset tabular dan mampu menangani banyak fitur hasil encoding. Random Forest juga menyediakan informasi mengenai fitur apa yang paling berpengaruh terhadap prediksi.

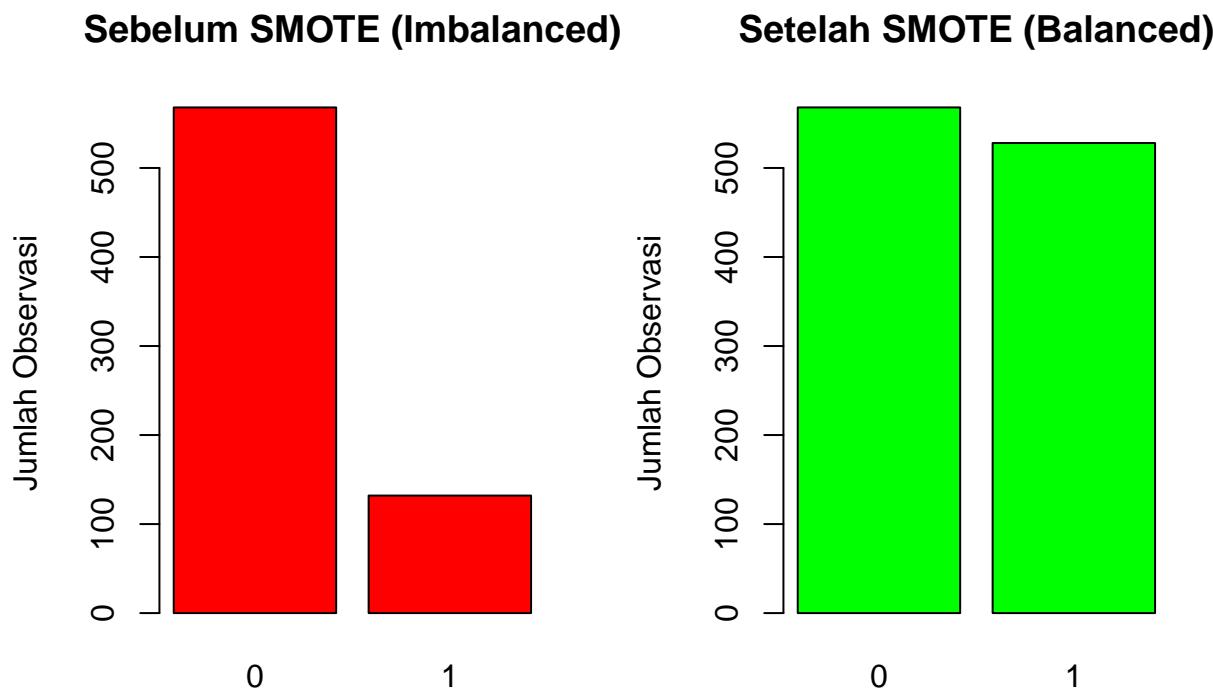
5.4 visualisasi data sebelum dan sesudah menggunakan SMOTE

```

par(mfrow=c(1,2))
barplot(table(train_data$churned),
        main="Sebelum SMOTE (Imbalanced)", col="red",
        ylab="Jumlah Observasi")

barplot(table(train_smote$churned),
        main="Setelah SMOTE (Balanced)", col="green",
        ylab="Jumlah Observasi")

```



```
par(mfrow=c(1,1))
```

pada visualisasi ini menunjukkan perubahan kelas sebelum dan sesudah SMOTE

6 Evaluation

```
# Prediksi menggunakan test_data (yang sudah di-encoded)
pred <- predict(rf_model, test_data)

# Tampilkan Confusion Matrix
# Menggunakan mode = "prec_recall" (sudah diperbaiki case sensitivity-nya)
cat("\nHasil Evaluasi (Confusion Matrix & Metrik):")
```

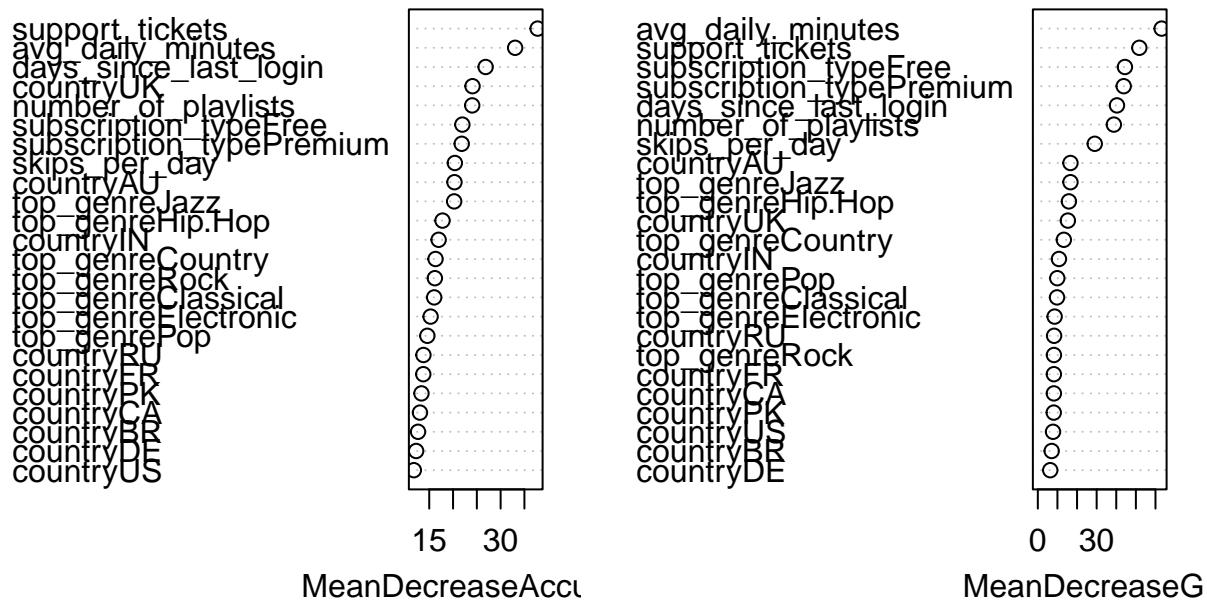
```
## 
## Hasil Evaluasi (Confusion Matrix & Metrik):

confusionMatrix(pred, test_data$churned, positive = "1", mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
```

```
##  
##          Reference  
## Prediction 0 1  
##          0 234 36  
##          1 12 18  
##  
##          Accuracy : 0.84  
##          95% CI : (0.7935, 0.8796)  
## No Information Rate : 0.82  
## P-Value [Acc > NIR] : 0.2056826  
##  
##          Kappa : 0.3443  
##  
## McNemar's Test P-Value : 0.0009009  
##  
##          Precision : 0.6000  
##          Recall : 0.3333  
##          F1 : 0.4286  
##          Prevalence : 0.1800  
## Detection Rate : 0.0600  
## Detection Prevalence : 0.1000  
## Balanced Accuracy : 0.6423  
##  
## 'Positive' Class : 1  
##  
  
# Cek Variable Importance (Fitur yang paling berpengaruh)  
varImpPlot(rf_model, main = "Variable Importance Random Forest")
```

Variable Importance Random Forest



pada evaluasi ini hasil menunjukkan model berhasil mencapai akurasi sebesar 0.84 yang berarti 84% prediksi model sudah sesuai dengan kondisi aktual pengguna. Namun, nilai recall untuk kelas churn (0.3333) menunjukkan bahwa model masih kesulitan dalam mendeteksi seluruh pengguna yang benar-benar churn.

analisa dari metrik krusial menunjukkan

- Tingkat Kegagalan Deteksi (Recall): Nilai Recall model ini sangat rendah, yaitu hanya 33.33%.
- Nilai Prediksi tercatat sebanyak 60%

Berdasarkan grafik, fitur yang memiliki pengaruh paling besar adalah avg_daily_minutes, support_tickets, subscription_type, dan days_since_last_login. Hal ini menunjukkan bahwa waktu penggunaan harian, frekuensi pengguna menghubungi layanan dukungan, jenis langganan yang digunakan, dan lama sejak terakhir login menjadi faktor penting dalam membedakan pengguna yang churn dan yang tetap aktif.

7 Kesimpulan dan saran

7.1 Kesimpulan

berdasarkan hasil dari penelitian ini dapat disimpulkan bahwa saat ini model berhasil menghasilkan akurasi sebesar 84% namun analisa dari matrix yang lebih mendalam menunjukkan bahwa daya prediksi recall buruk sebesar 33.33% dan hasil precision sebesar 60%, oleh karena itu model ini belum bisa diimplementasikan untuk proses bisnis karena bisa megakibatkan kerugian akibat recall yang tergolong kecil

7.2 saran

untuk penelitian selanjutnya disarankan:

- perlu difokuskan untuk meningkatkan nilai dari recall kedepannya
- minimnya informasi yang didapat dari dataset oleh sebab itu perlu menambah fitur yang adapada dataset
- Memperbaiki kualitas data untuk mengurangi nilai yang tidak logis