Customer Churn Prediction using Machine Learning

by Marcel Siswanto – 24 July 2020

I. Introduction

Latar Belakang

DQLab Telco merupakan perusahaan Telco yang sudah mempunyai banyak cabang tersebar dimana-mana. Berdiri sejak tahun 2019, DQLab Telco konsisten untuk memperhatikan customer experience sehingga tidak akan ditinggalkan pelanggan.

Walaupun baru berumur 1 tahun lebih sedikit, DQLab Telco sudah mempunyai banyak pelanggan yang beralih langganan ke kompetitior. Pihak management ingin mengurangi jumlah pelanggan yang beralih (churn) dengan menggunakan machine learning.

Pada project kali ini, akan diprediksi churn pelanggan.

Tugas dan Langkah

Sebagai data scientist, saya diminta untuk membuat model yang tepat.

Pada project kali ini, saya akan melakukan permodelan machine learning dengan menggunakan data bulan lalu, yakni Juni 2020.

Langkah yang akan dilakukan adalah:

1. Melakukan Exploratory Data Analysis
2. Melakukan Data Pre-Processing
3. Melakukan Permodelan Machine Learning
4. Menentukan Model Terbaik

II. Library dan Data yang Digunakan

Library yang Digunakan

Pada analisis kali ini, akan digunakan beberapa package yang membantu kita dalam melakukan analisis data,

1. Pandas (Python for Data Analysis) adalah library Python yang berfokus untuk proses analisis data seperti manipulasi data, persiapan data, dan pembersihan data.
   - `read_csv()` digunakan untuk membaca file csv
   - `replace()` digunakan untuk mengganti nilai
   - `value_counts()` digunakan untuk mengitung unik dari kolom
   - `drop()` digunakan untuk menghapus
   - `describe()` digunakan untuk melihat deskripsi datanya
   - `value_counts()` digunakan untuk mengitung unik dari kolom
2. Matplotlib adalah library Python yang fokus pada visualisasi data seperti membuat plot grafik. Matplotlib dapat digunakan dalam skrip Python, Python dan IPython shell, server aplikasi web, dan beberapa toolkit graphical user interface (GUI) lainnya.
   - `figure()` digunakan untuk membuat figure gambar baru
   - `subplots()` digunakan untuk membuat gambar dan satu set subplot
   - `title()` digunakan untuk memberi judul pada gambar
   - `ylabel()` digunakan untuk memberi label sumbu Y pada gambar
   - `xlabel()` digunakan untuk memberi label sumbu Y pada gambar
   - `pie()` digunakan untuk membuat pie chart
3. Seaborn membangun plot di atas Matplotlib dan memperkenalkan tipe plot tambahan. Ini juga membuat plot Matplotlib tradisional Anda terlihat lebih cantik.
   - `countplot()` digunakan untuk membuat plot dengan jumlah pengamatan di setiap bin kategorik variable
   - `heatmap()` Plot rectangular data as a color-encoded matrix
4. Scikit-learn adalah library dalam Python yang menyediakan banyak algoritma Machine Learning baik untuk Supervised, Unsupervised Learning, maupun digunakan untuk mempreparasi data.
   - `LabelEncoder()` digunakan untuk merubah nilai dari suatu variable menjadi 0 atau 1
   - `train_test_split()` digunakan untuk membagi data menjadi 2 row bagian (Training & Testing)
   - `LogisticRegression()` digunakan untuk memanggil algoritma Logistic Regression
   - `RandomForestClassifier()` digunakan untuk memanggil algoritma Random Forest Classifier

- confusion_matrix() digunakan untuk membuat confusion matrix
- classification_report() digunakan untuk membuat classification report, yang dianataranya berisi akurasi model

5. Xgboost adalah library dalam Python untuk algoritma extreme gradient boosting (xgboost)
   - XGBClassifier() digunakan untuk memanggil algoritma XG Boost Classifier
6. Pickle mengimplementasikan protokol biner untuk serializing dan de-serializing dari struktur objek Python.
   - dump() digunakan untuk menyimpan

Import Library yang Dibutuhkan

```
In [1]: #Importing general packages
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.ensemble import GradientBoostingClassifier
        from sklearn.metrics import confusion_matrix, classification_report
        import pickle
        from pathlib import Path
```

Data yang Digunakan

Untuk Dataset yang digunakan sudah disediakan dalam format csv, silahkan baca melalui fungsi pandas di python:

df_load = pd.read_csv('https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/dqlab_telco_final.csv')

Untuk detail datanya adalah sebagai berikut:

- UpdatedAt Periode of Data taken
- customerID Customer ID
- gender Whether the customer is a male or a female (Male, Female)
- SeniorCitizen Whether the customer is a senior citizen or not (Yes, No)
- Partner Whether the customer has a partner or not (Yes, No)
- tenure Number of months the customer has stayed with the company
- PhoneService Whether the customer has a phone service or not (Yes, No)
- InternetService Customer's internet service provider (Yes, No)

- StreamingTV Whether the customer has streaming TV or not (Yes, No)
- PaperlessBilling Whether the customer has paperless billing or not (Yes, No)
- MonthlyCharges The amount charged to the customer monthly
- TotalCharges The total amount charged to the customer
- Churn Whether the customer churned or not (Yes, No)

File Unloading

Lakukan import dataset ke dalam workspace dengan menggunakan read_csv dan tampilkan juga bentuk atau shape dari dataset tersebut beserta 5 data teratas.

Sumber dataset : https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/dqlab_telco_final.csv

In [2]:
```
#Import dataset
df_load = pd.read_csv('https://dqlab-dataset.s3-ap-southeast-1.amazonaws.com/dqlab_telco_final.csv')
#Tampilkan bentuk dari dataset
print(df_load.shape)
#Tampilkan 5 data teratas
print(df_load.head())
#Tampilkan jumlah ID yang unik
print(df_load.customerID.nunique())
```

```
(6950, 13)
   UpdatedAt   customerID  gender SeniorCitizen Partner  tenure PhoneService  \
0     202006  45759018157  Female            No     Yes       1           No
1     202006  45315483266    Male            No     Yes      60          Yes
2     202006  45236961615    Male            No      No       5          Yes
3     202006  45929827382  Female            No     Yes      72          Yes
4     202006  45305082233  Female            No     Yes      56          Yes
```

```
  StreamingTV InternetService PaperlessBilling  MonthlyCharges  Tot
alCharges  \
0          No             Yes               Yes           29.85
29.85
1          No              No               Yes           20.50
1198.80
2         Yes             Yes                No          104.10
541.90
3         Yes             Yes               Yes          115.50
8312.75
4         Yes             Yes                No           81.25
4620.40

  Churn
0    No
1    No
2   Yes
3    No
4    No
6950
```

III. Melakukan Exploratory Data Analysis (EDA)

Exploratory Data Analysis

Exploratory Data Analysis memungkinkan analyst memahami isi data yang digunakan, mulai dari distribusi, frekuensi, korelasi dan lainnya. Pada umumnya EDA dilakukan dengan beberapa cara:

- Univariat Analysis — analisis deskriptif dengan satu variabel.
- Bivariat Analysis — analisis relasi dengan dua variabel yang biasanya dengan target variabel.
- Multivariat Analysis — analisis yang menggunakan lebih dari atau sama dengan tiga variabel.

Dalam kasus ini, kita diminta untuk melihat persebaran dari:

- Prosentase persebaran data Churn dan tidaknya dari seluruh data
- Persebarang data dari variable predictor terhadap label (Churn)

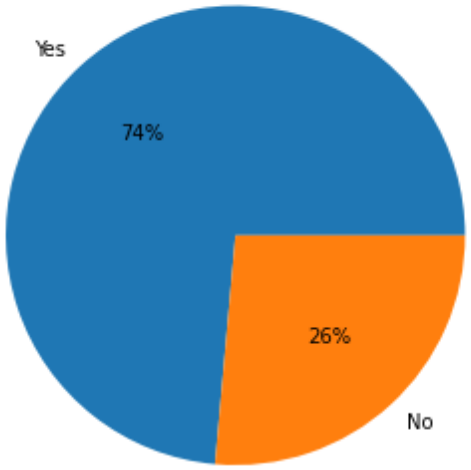
In [3]:
```
#Import matplotlib and seaborn
import matplotlib.pyplot as plt
import seaborn as sns
```

Memvisualisasikan Prosentase Churn

Kita ingin melihat visualisasi data secara univariat terkait prosentase data churn dari pelanggan. Gunakan fungsi value_counts() untuk menghitung banyaknya unik dari sebuah kolom, pie() untuk membuat pie chart.

In [4]:
```
from matplotlib import pyplot as plt
import numpy as np

#Your codes here
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
labels = ['Yes','No']
churn = df_load.Churn.value_counts()
ax.pie(churn, labels=labels, autopct='%.0f%%')
plt.show()
```



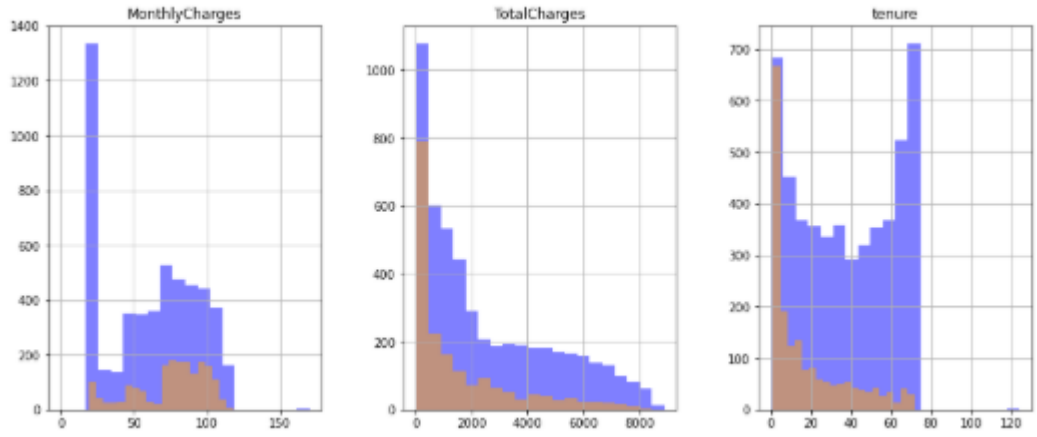
Exploratory Data Analysis (EDA) Variabel Numerik

Hal yang akan kita lakukan selanjutnya adalah memilih variable predictor yang bersifat numerik dan membuat plot secara bivariat, kemudian menginterpretasikannya

Gunakan data `df_load` untuk diolah di tahap ini dan gunakan fungsi `subplots()` untuk membuat gambar dan satu set subplot.

In [5]:
```
from matplotlib import pyplot as plt
import numpy as np

#Creating bin in chart
numerical_features = ['MonthlyCharges','TotalCharges','tenure']
fig, ax = plt.subplots(1, 3, figsize=(15, 6))
#Use the following code to plot two overlays of histogram per each
numerical_features, use a color of blue and orange, respectively
df_load[df_load.Churn == 'No'][numerical_features].hist(bins=20, co
lor='blue', alpha=0.5, ax=ax)
df_load[df_load.Churn == 'Yes'][numerical_features].hist(bins=20, c
olor='orange', alpha=0.5, ax=ax)
plt.show()
```



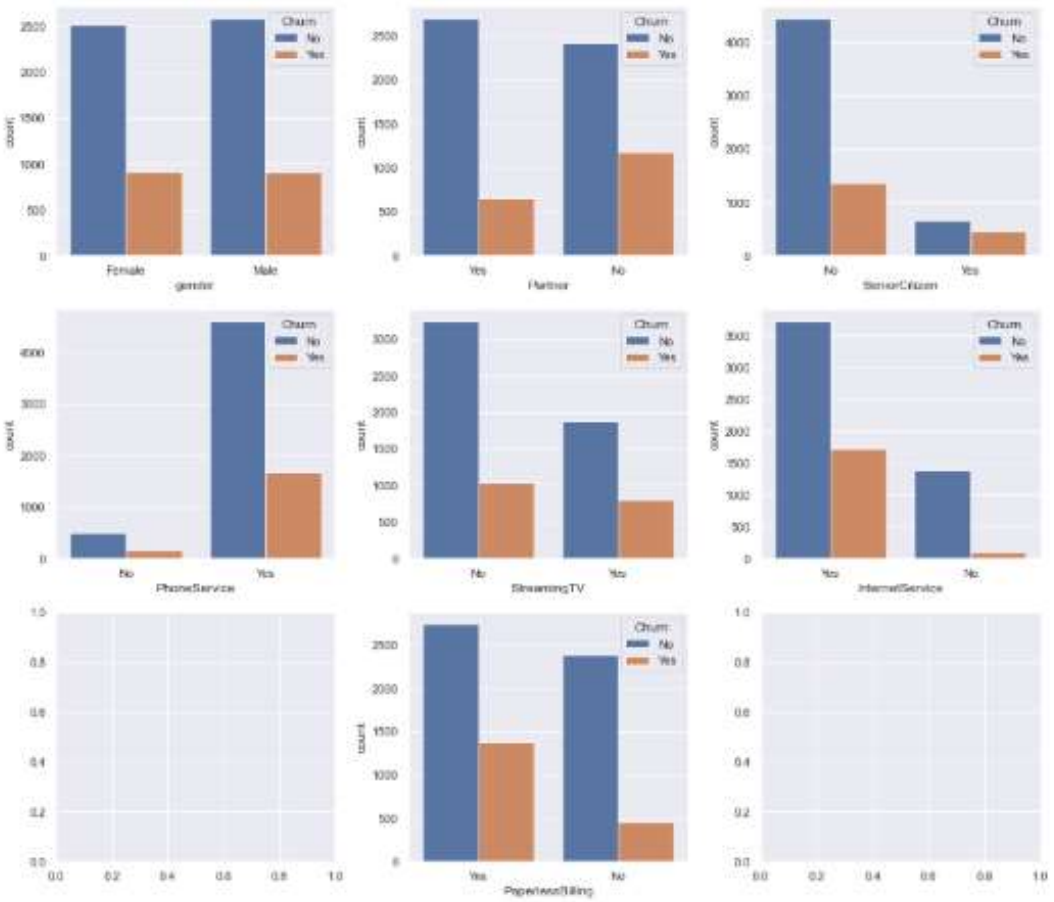
Exploratory Data Analysis (EDA) Variabel Kategorik

Setelah itu, kita akan melakukan pemilihan variable predictor yang bersifat kategorik dan membuat plot secara bivariat, kemudian menginterpretasikannya.

Gunakan data `df_load` untuk diolah di tahap ini. Gunakan fungsi `countplot()` untuk membuat plot dengan jumlah pengamatan di setiap bin kategorik variable.

In [6]:
```
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
sns.set(style='darkgrid')

#Your code goes here
fig, ax = plt.subplots(3, 3, figsize=(14, 12))
sns.countplot(data=df_load, x='gender', hue='Churn', ax=ax[0][0])
sns.countplot(data=df_load, x='Partner', hue='Churn', ax=ax[0][1])
sns.countplot(data=df_load, x='SeniorCitizen', hue='Churn', ax=ax
[0][2])
sns.countplot(data=df_load, x='PhoneService', hue='Churn', ax=ax[1]
[0])
sns.countplot(data=df_load, x='StreamingTV', hue='Churn', ax=ax[1]
[1])
sns.countplot(data=df_load, x='InternetService', hue='Churn', ax=ax
[1][2])
sns.countplot(data=df_load, x='PaperlessBilling', hue='Churn', ax=a
x[2][1])
plt.tight_layout()
plt.show()
```



Kesimpulan

Berdasarkan hasil dan analisa di atas dapat disimpulkan:

- Pada tahap C.1 dapat kita ketahui bahwa sebaran data secara kesuluruhan customer tidak melakukan churn, dengan detil Churn sebanyak 26% dan No Churn sebanyak 74%.
- Pada tahap C.2 dapat kita ketahui bahwa untuk MonthlyCharges ada kecenderungan semakin kecil nilai biaya bulanan yang dikenakan, semakin kecil juga kecenderungan untuk melakukan Churn. Untuk TotalCharges terlihat tidak ada kecenderungan apapun terhadap Churn customers. Untuk tenure ada kecenderungan semakin lama berlangganan customer, semakin kecil kecenderungan untuk melakukan Churn.
- Pada tahap C.3 dapat kita ketahui bahwa tidak ada perbedaan yang signifikan untuk orang melakukan churn dilihat dari faktor jenis kelamin (gender) dan layanan telfonnya (PhoneService). Akan tetapi ada kecenderungan bahwa orang yang melakukan churn adalah orang-orang yang tidak memiliki partner (partner: No), orang-orang yang statusnya adalah senior citizen(SeniorCitizen: Yes), orang-orang yang mempunyai layanan streaming TV (StreamingTV: Yes), orang-orang yang mempunyai layanan Internet (internetService: Yes) dan orang-orang yang tagihannya paperless (PaperlessBilling: Yes).

IV. Melakukan Data Pre-processing

Menghapus Unnecessary Columns dari Data

Selanjutnya, kita akan mengapus kolom yang tidak akan diikutsertakan dalam pemodelan, kemudian simpan dengan nama cleaned_df. Tampilkan 5 rows teratasnya.

Gunakan drop() untuk menghapus kolom dari suatu data.

```
In [7]: #Remove the unnecessary columns customerID & UpdatedAt
        cleaned_df = df_load.drop(['customerID', 'UpdatedAt'], axis=1)
        print(cleaned_df.head())
```

```
   gender SeniorCitizen Partner  tenure PhoneService StreamingTV  \
0  Female             No     Yes       1           No          No
1    Male             No     Yes      60          Yes          No
2    Male             No      No       5          Yes         Yes
3  Female             No     Yes      72          Yes         Yes
4  Female             No     Yes      56          Yes         Yes

  InternetService PaperlessBilling  MonthlyCharges  TotalCharges Ch
urn
0             Yes              Yes           29.85         29.85
No
1              No              Yes           20.50       1198.80
No
2             Yes               No          104.10        541.90
Yes
3             Yes              Yes          115.50       8312.75
No
4             Yes               No           81.25       4620.40
No
```

Encoding Data

Gunakan data dari hasil dan analisa sebelumnya cleaned_df, untuk merubah value dari data yang masih berbentuk string untuk diubah ke dalam bentuk numeric menggunakan LabelEncoder().

Gunakan describe() untuk melihat deskripsi datanya.

In [8]:
```
from sklearn.preprocessing import LabelEncoder
#Convert all the non-numeric columns to numerical data types
for column in cleaned_df.columns:
    if cleaned_df[column].dtype == np.number: continue
    # Perform encoding for each non-numeric column
    cleaned_df[column]  = LabelEncoder().fit_transform(cleaned_df[c
olumn])
print(cleaned_df.describe())
```

```
            gender  SeniorCitizen      Partner       tenure  PhoneS
ervice  \
count  6950.000000    6950.000000  6950.000000  6950.000000   6950.
000000
mean      0.504317       0.162302     0.483309    32.415827      0.
903741
std       0.500017       0.368754     0.499757    24.561336      0.
294967
min       0.000000       0.000000     0.000000     0.000000      0.
000000
25%       0.000000       0.000000     0.000000     9.000000      1.
000000
50%       1.000000       0.000000     0.000000    29.000000      1.
000000
75%       1.000000       0.000000     1.000000    55.000000      1.
000000
max       1.000000       1.000000     1.000000    73.000000      1.
000000

       StreamingTV  InternetService  PaperlessBilling  MonthlyCharg
es  \
count  6950.000000      6950.000000       6950.000000     6950.0000
00
mean      0.384317         0.783453          0.591942       64.9922
01
std       0.486468         0.411921          0.491509       30.0320
40
min       0.000000         0.000000          0.000000        0.0000
00
25%       0.000000         1.000000          0.000000       36.4625
00
50%       0.000000         1.000000          1.000000       70.4500
00
75%       1.000000         1.000000          1.000000       89.8500
00
max       1.000000         1.000000          1.000000      169.9312
50
```

```
       TotalCharges        Churn
count   6950.000000  6950.000000
mean    2286.058750     0.264173
std     2265.702553     0.440923
min       19.000000     0.000000
25%      406.975000     0.000000
50%     1400.850000     0.000000
75%     3799.837500     1.000000
max     8889.131250     1.000000
```

Splitting Dataset

Gunakan data dari hasil dan analisa sebelumnya cleaned_df, untuk dibagi datasetnya menjadi 2 bagian (70% training & 30% testing) berdasarkan variable predictor (X) dan targetnya (Y). Gunakan train_test_split() untuk membagi data tersebut. Sertakan value_counts untuk mengecek apakah pembagian sudah sama proporsinya. Simpan hasil spliting data menjadi x_train, y_train, x_test & y_test.

In [9]:
```
from sklearn.model_selection import train_test_split

#Predictor dan target
x = cleaned_df.drop('Churn', axis=1)
y = cleaned_df['Churn']

#Splitting train and test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size
=0.3, random_state=42)

#Print according to the expected result
print('Jumlah baris dan kolom dari x_train adalah:', x_train.shap
e,', sedangkan Jumlah baris dan kolom dari y_train adalah:', y_trai
n.shape)
print('Prosentase Churn di data Training adalah:')
print(y_train.value_counts(normalize=True))
print('Jumlah baris dan kolom dari x_test adalah:', x_test.shape,',
sedangkan Jumlah baris dan kolom dari y_test adalah:', y_test.shap
e)
print('Prosentase Churn di data Testing adalah:')
print(y_test.value_counts(normalize=True))
```

```
Jumlah baris dan kolom dari x_train adalah: (4865, 10) , sedangkan
Jumlah baris dan kolom dari y_train adalah: (4865,)
Prosentase Churn di data Training adalah:
0    0.734841
1    0.265159
Name: Churn, dtype: float64
Jumlah baris dan kolom dari x_test adalah: (2085, 10) , sedangkan J
umlah baris dan kolom dari y_test adalah: (2085,)
Prosentase Churn di data Testing adalah:
0    0.738129
1    0.261871
Name: Churn, dtype: float64
```

Kesimpulan

Setelah kita analisis lebih lanjut, ternyata ada kolom yang tidak dibutuhkuan dalam model, yaitu Id Number pelanggannya (customerID) & periode pengambilan datanya (UpdatedAt), maka hal ini perlu dihapus. Kemudian kita lanjut mengubah value dari data yang masih berbentuk string menjadi numeric melalui encoding, setelah dilakukan terlihat di persebaran datanya khususnya kolom min dan max dari masing masing variable sudah berubah menjadi 0 & 1. Tahap terakhir adalah membagi data menjadi 2 bagian untuk keperluan modelling, setelah dilakukan terlihat dari jumlah baris dan kolom masing-masing data sudah sesuai & prosentase kolom churn juga sama dengan data di awal, hal ini mengindikasikan bahwasannya data terpisah dengan baik dan benar.

V. Modelling: Logistic Regression

Pembuatan Model

Selanjutnya kita akan membuat model dengan menggunakan Algoritma Logistic Regression. Gunakan `LogisticRegression()` memanggil algoritma tersebut, fit ke data train dan simpan sebagai `log_model`

In [10]:
```
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression().fit(x_train, y_train)
print('Model Logistic Regression yang terbentuk adalah:', log_mode
l)
```

```
Model Logistic Regression yang terbentuk adalah: LogisticRegression
()
```

Performansi Model Training – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data training seperti hasil di bawah ini.

Gunakan `classification_report()` & `confusion_matrix()`.

In [11]:
```
from sklearn.metrics import classification_report
#Predict
y_train_pred = log_model.predict(x_train)
#Print classification report
print('Classification Report Training Model (Logistic Regression)
:')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Logistic Regression) :
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      3575
           1       0.65      0.50      0.56      1290

    accuracy                           0.80      4865
   macro avg       0.74      0.70      0.72      4865
weighted avg       0.79      0.80      0.79      4865
```

Performansi Model Training – Menampilkan Plots

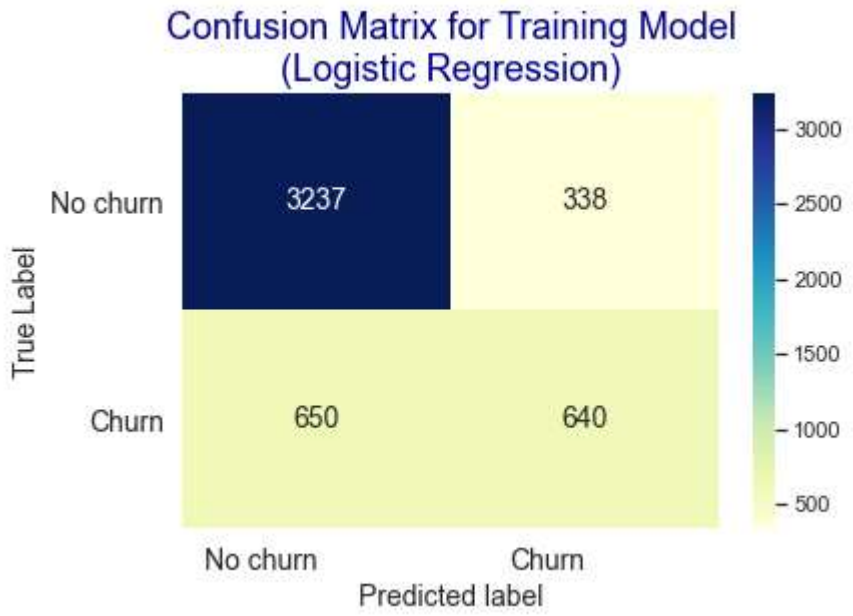
Setelah mendapatkan hasil classification report pada tahap sebelumnya, sekarang kita akan melakukan visualisasi terhadap report tersebut.

In [12]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_train, y_tra
in_pred)), ('No churn', 'Churn'), ('No churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Training Model\n(Logistic Regressio
n)', fontsize=18, color='darkblue')
plt.ylabel('True Label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Performansi Data Testing – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data testing seperti hasil di bawah ini. Gunakan classification_report() & confusion_matrix().

In [13]:
```
from sklearn.metrics import classification_report

#Predict
y_test_pred = log_model.predict(x_test)
#Print classification report
print('Classification Report Testing Model (Logistic Regression):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Logistic Regression):
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1539
           1       0.64      0.48      0.55       546

    accuracy                           0.79      2085
   macro avg       0.74      0.69      0.71      2085
weighted avg       0.78      0.79      0.78      2085
```

Performansi Data Testing – Menampilkan Plots

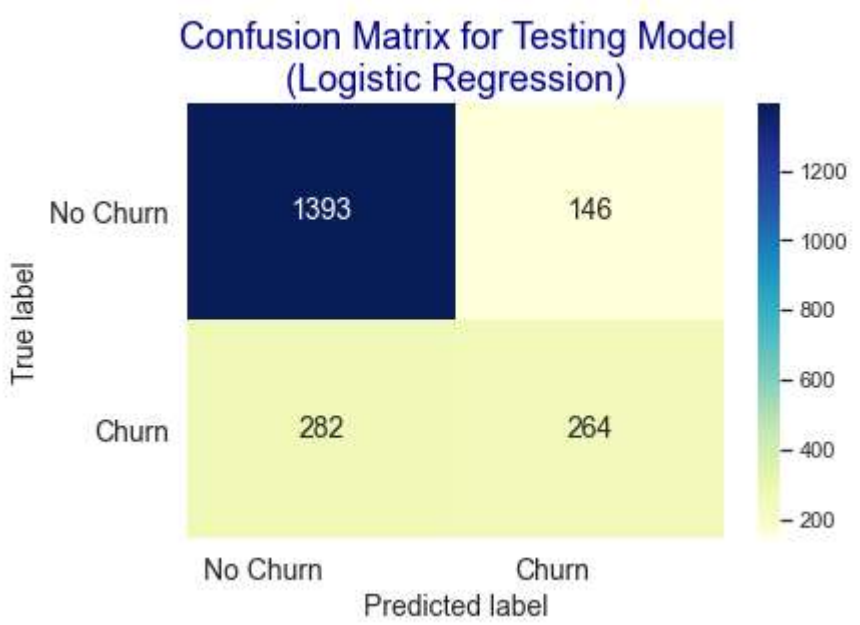
Setelah menampilkan metrics pada tahap sebelumnya, sekarang kita akan melakukan visualisasi dari metrics yang sudah dihasilkan sebelumnya.

In [14]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_test, y_test
_pred)), ('No Churn', 'Churn'), ('No Churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Testing Model\n(Logistic Regressio
n)', fontsize=18, color='darkblue')
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Kesimpulan

Dari hasil dan analisa di atas, maka:

- Jika kita menggunakan menggunakan algoritma logistic regression dengan memanggil LogisticRegression() dari sklearn tanpa menambahi parameter apapun, maka yang dihasilkan adalah model dengan seting default dari sklearn, untuk detilnya bisa dilihat pada dokumentasinya.
- Dari data training terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 80%, dengan detil tebakan churn yang sebenernya benar churn adalah 638, tebakan tidak churn yang sebenernya tidak churn adalah 3237, tebakan tidak churn yang sebenernya benar churn adalah 652 dan tebakan churn yang sebenernya tidak churn adalah 338.
- Dari data testing terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 79%, dengan detil tebakan churn yang sebenernya benar churn adalah 264, tebakan tidak churn yang sebenernya tidak churn adalah 1392, tebakan tidak churn yang sebenernya benar churn adalah 282 dan tebakan churn yang sebenernya tidak churn adalah 146.

VI. Modelling: Random Forest Classifier

Pembuatan Model

Selanjutnya kita akan membuat model dengan menggunakan Algoritma Random Forest Classifier.

Gunakan `RandomForestClassifier()` memanggil algoritma tersebut, fit ke data train dan simpan sebagai `rdf_model`

In [15]:
```
from sklearn.ensemble import RandomForestClassifier
#Train the model
rdf_model = RandomForestClassifier().fit(x_train, y_train)
print(rdf_model)
```

```
RandomForestClassifier()
```

Perfomansi Data Training – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data training seperti hasil di bawah ini.

Gunakan `classification_report()` & `confusion_matrix()`.

In [16]:
```
from sklearn.metrics import classification_report

#Predict
y_train_pred = rdf_model.predict(x_train)
#Print Classification Report
print('Classification Report Training Model (Random Forest) :')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Random Forest) :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      3575
           1       0.99      0.99      0.99      1290

    accuracy                           1.00      4865
   macro avg       0.99      0.99      0.99      4865
weighted avg       1.00      1.00      1.00      4865
```

Performansi Data Training – Menampilkan Plots

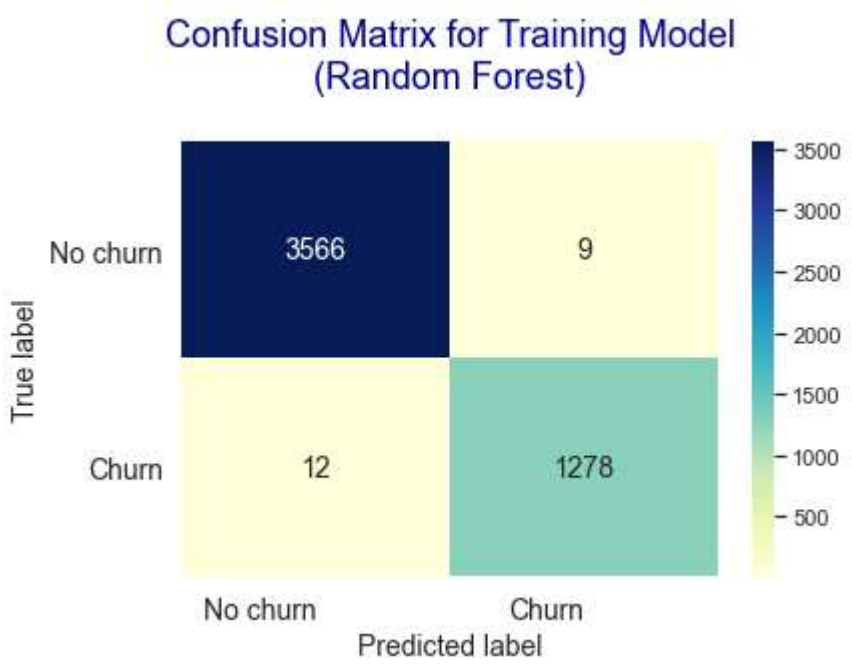
Setelah menampilkan metrics pada tahap sebelumnya, selanjutnya kita akan melakukan visualisasi terhadap metrics tersebut.

In [17]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_train, y_tra
in_pred)), ('No churn', 'Churn'), ('No churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Training Model\n(Random Forest)\n',
fontsize=18, color='darkblue')
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Performansi Data Training – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data testing seperti hasil di bawah ini. Gunakan classification_report() & confusion_matrix().

In [18]:
```
from sklearn.metrics import classification_report

# Predict
y_test_pred = log_model.predict(x_test)
# Print classification report
print('Classification Report Testing Model (Random Forest Classifie
r):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Random Forest Classifier):
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1539
           1       0.64      0.48      0.55       546

    accuracy                           0.79      2085
   macro avg       0.74      0.69      0.71      2085
weighted avg       0.78      0.79      0.78      2085
```

Performansi Data Testing – Menampilkan Plots

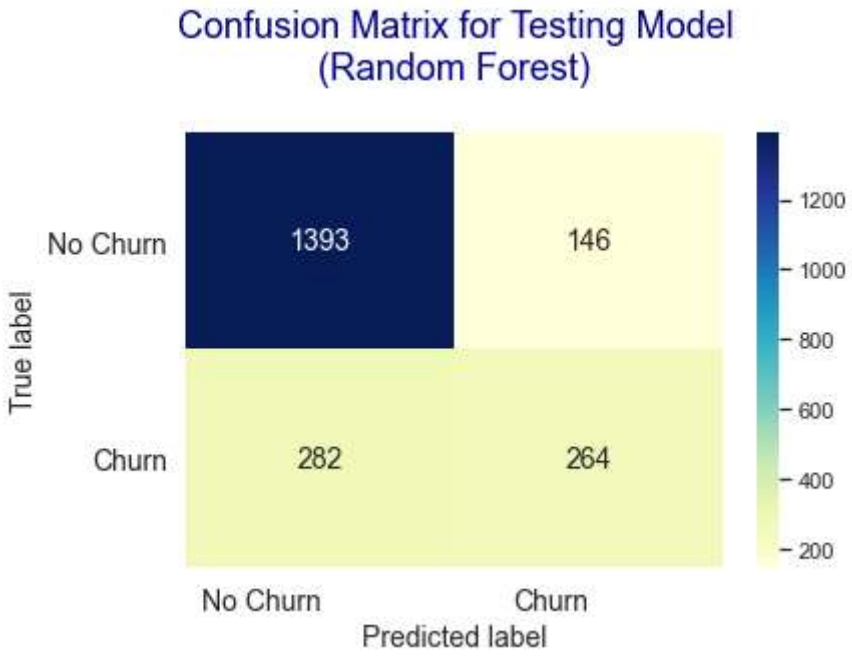
Tampilkan visualisasi dari hasil metrics yang sudah diperoleh pada tahap sebelumnya.

In [19]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_test, y_test
_pred)), ('No Churn', 'Churn'), ('No Churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Testing Model\n(Random Forest)\n',
fontsize=18, color='darkblue')
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Kesimpulan

Dari hasil dan analisa di atas, maka:

- Jika kita menggunakan algoritma Random Forest dengan memanggil RandomForestClassifier() dari sklearn tanpa menambahi parameter apapun, maka yang dihasilkan adalah model dengan setting default dari sklearn, untuk detilnya bisa dilihat pada dokumentasinya.
- Dari data training terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 100%, dengan detil tebakan churn yang sebenarnya benar churn adalah 1278, tebakan tidak churn yang sebenarnya tidak churn adalah 3566, tebakan tidak churn yang sebenarnya benar churn adalah 12 dan tebakan churn yang sebenarnya tidak churn adalah 9.
- Dari data testing terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 78%, dengan detil tebakan churn yang sebenarnya benar churn adalah 264, tebakan tidak churn yang sebenarnya tidak churn adalah 1393, tebakan tidak churn yang sebenarnya benar churn adalah 282 dan tebakan churn yang sebenarnya tidak churn adalah 146.

VII. Modelling: Gradient Boosting Classifier

Pembuatan Model

Selanjutnya kita akan membuat model dengan menggunakan Algoritma Gradient Boosting Classifier.

Gunakan GradientBoostingClassifier() memanggil algoritma tersebut, fit ke data train dan simpan sebagai gbt_model

In [20]:
```
from sklearn.ensemble import GradientBoostingClassifier
#Train the model
gbt_model = GradientBoostingClassifier().fit(x_train, y_train)
print(gbt_model)
```

```
GradientBoostingClassifier()
```

Performansi Model Data Training – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data training seperti hasil di bawah ini. Gunakan classification_report() & confusion_matrix().

In [21]:
```
from sklearn.metrics import classification_report
# Predict
y_train_pred = gbt_model.predict(x_train)
# Print classification report
print('Classification Report Training Model (Gradient Boosting):')
print(classification_report(y_train, y_train_pred))
```

```
Classification Report Training Model (Gradient Boosting):
              precision    recall  f1-score   support

           0       0.84      0.92      0.88      3575
           1       0.70      0.53      0.60      1290

    accuracy                           0.82      4865
   macro avg       0.77      0.72      0.74      4865
weighted avg       0.81      0.82      0.81      4865
```

Performansi Model Data Training – Menampilkan Plots

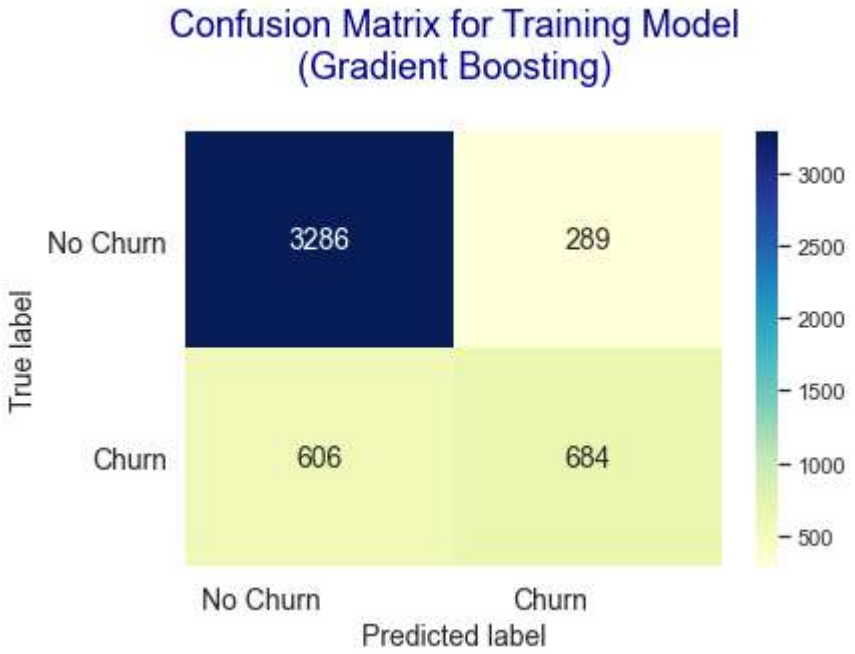
Tampilkan visualisasi dari metrics yang sudah dihasilkan sebelumnya.

In [22]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_train, y_tra
in_pred)), ('No Churn', 'Churn'), ('No Churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Training Model\n(Gradient Boosting)
\n', fontsize=18, color='darkblue')
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Performansi Model Data Testing – Menampilkan Metrics

Setelah kita membuat modelnya, maka lakukan perhitungan untuk memperoleh classification reportnya dan confusion matrixnya di data testing seperti hasil di bawah ini. Gunakan `classification_report()` & `confusion_matrix()`.

In [23]:
```
from sklearn.metrics import classification_report
# Predict
y_test_pred = log_model.predict(x_test)
# Print classification report
print('Classification Report Testing Model (Gradient Boosting):')
print(classification_report(y_test, y_test_pred))
```

```
Classification Report Testing Model (Gradient Boosting):
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1539
           1       0.64      0.48      0.55       546

    accuracy                           0.79      2085
   macro avg       0.74      0.69      0.71      2085
weighted avg       0.78      0.79      0.78      2085
```

Performansi Model Data Testing – Menampilkan Plots

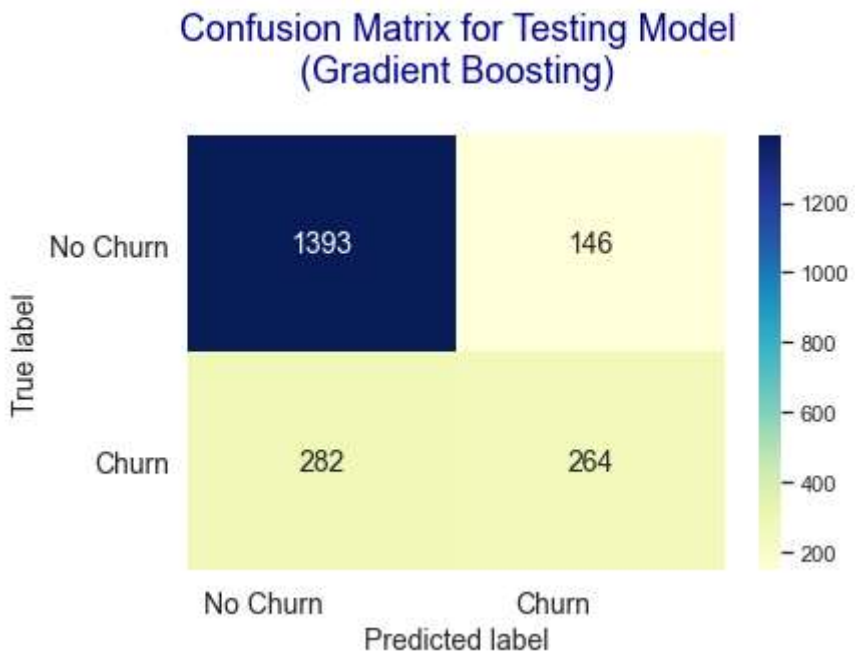
Buatlah visualisasi dari metrics yang sudah dihasilkan sebelumnya.

In [24]:
```
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns

# Form confusion matrix as a DataFrame
confusion_matrix_df = pd.DataFrame((confusion_matrix(y_test, y_test
_pred)), ('No Churn', 'Churn'), ('No Churn', 'Churn'))

# Plot confusion matrix
plt.figure()
heatmap = sns.heatmap(confusion_matrix_df, annot=True, annot_kws=
{'size': 14}, fmt='d', cmap='YlGnBu')
heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotati
on=0, ha='right', fontsize=14)

plt.title('Confusion Matrix for Testing Model\n(Gradient Boosting)
\n', fontsize=18, color='darkblue')
plt.ylabel('True label', fontsize=14)
plt.xlabel('Predicted label', fontsize=14)
plt.show()
```



Kesimpulan

Dari hasil dan analisa di atas, maka:

- Jika kita menggunakan menggunakan algoritma Gradient Boosting dengan memanggil GradientBoostingClassifier() dari package sklearn tanpa menambahi parameter apapun, maka yang dihasilkan adalah model dengan setting default dari sklearn, untuk detilnya bisa dilihat pada dokumentasinya.
- Dari data training terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 82%, dengan detil tebakan churn yang sebenarnya benar churn adalah 684, tebakan tidak churn yang sebenarnya tidak churn adalah 3286, tebakan tidak churn yang sebenarnya benar churn adalah 606 dan tebakan churn yang sebenarnya tidak churn adalah 289.
- Dari data testing terlihat bahwasannya model mampu memprediksi data dengan menghasilkan akurasi sebesar 79%, dengan detil tebakan churn yang sebenarnya benar churn adalah 264, tebakan tidak churn yang sebenarnya tidak churn adalah 1393, tebakan tidak churn yang sebenarnya benar churn adalah 282 dan tebakan churn yang sebenarnya tidak churn adalah 146.

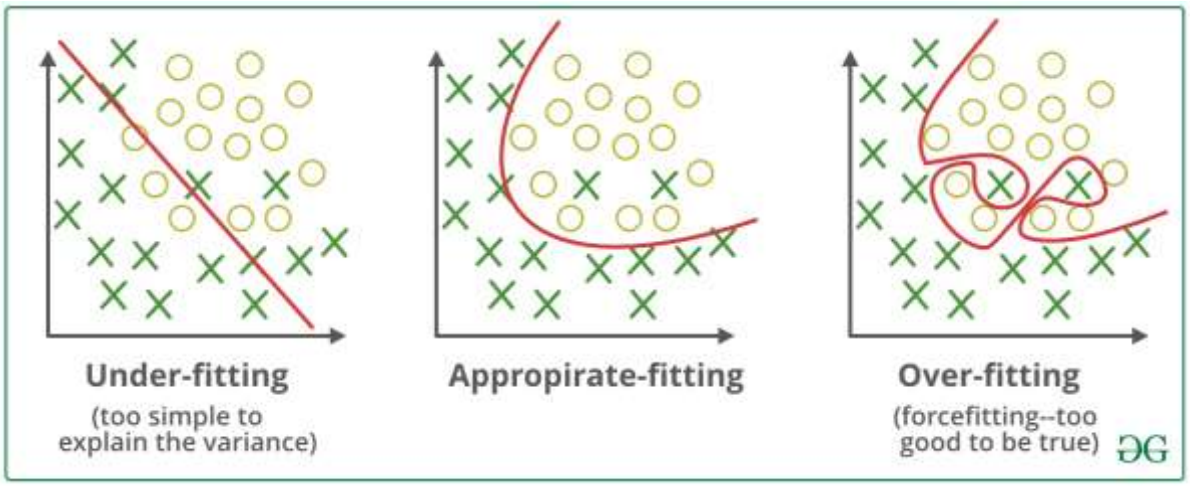
VIII. Memilih Model Terbaik

Menentukan Algoritma Model Terbaik

Model yang baik adalah model yang mampu memberikan performa bagus di fase training dan testing model.

- Over-Fitting adalah suatu kondisi dimana model mampu memprediksi dengan sangat baik di fase training, akan tetapi tidak mampu memprediksi sama baiknya di fase testing.
- Under-Fitting adalah suatu kondisi dimana model kurang mampu memprediksi dengan baik di fase training, akan tetapi mampu memprediksi dengan baik di fase testing.
- Appropriate-Fitting adalah suatu kondisi dimana model mampu memprediksi dengan baik di fase training maupun di fase testing.

Untuk detailnya, bisa dilihat ilustrasi di bawah ini:



Selanjutnya kita akan menentukan model algoritma terbaik dari model yang sudah di lakukan di atas (Appropriate-Fitting), kemudian kita simpan sebagai file best_model_churn.pkl dengan tujuan untuk deployment model nantinya kita tidak perlu mengulang lagi pemodelan, cukup memanggil file tersebut saja. Simpan di file direktori sesuai dataset berada, kemudian check apakah file tersebut benar tersimpan atau tidak. Gunakan dump() dari pickle untuk menyimpan file. Anda bisa gunakan code dibawah ini untuk menyimpan file model untuk di coba di local laptop pribadi Anda.

```
#Save Model
pickle.dump(log_model, open('best_model_churn.pkl', 'wb'))
```

```
In [25]: print(log_model)

         LogisticRegression()
```

Kesimpulan

Berdasarkan pemodelan yang telah dilakukan dengan menggunakan Logistic Regression, Random Forest dan Extreme Gradiant Boost, maka dapat disimpulkan untuk memprediksi churn dari pelanggan telco dengan menggunakan dataset ini model terbaiknya adalah menggunakan algortima Logistic Regression. Hal ini dikarenakan performa dari model Logistic Regression cenderung mampu memprediksi sama baiknya di fase training maupun testing (akurasi training 80%, akurasi testing 79%), dilain sisi algoritma lainnya cenderung Over-Fitting performanya. Akan tetapi hal ini tidak menjadikan kita untuk menarik kesimpulan bahwsannya jika untuk melakukan pemodelan apapun maka digunakan Logistic Regression, kita tetap harus melakukan banyak percobaan model untuk menentukan mana yang terbaik.