

Analisis Churn Prediction pada Perusahaan Telekomunikasi

Git Hub Link : https://github.com/axeltanjung/telecom_pred

Formulasi Masalah

Dengan perkembangan telekomunikasi industry yang cepat, *service provider* semakin menurun akibat ekspansi dari ke arah yang berbasis *subscriber*. Untuk memenuhi kebutuhan akan perusahaan atas lingkungan yang kompetitif, retensi dari *customer* eksisting menjadi tantangan yang besar. Hal ini menyatakan bahwa *cost* untuk mendapatkan *customer* baru lebih tinggi dibandingkan mempertahankan yang lama.

Maka dari itu, sebuah keniscayaan untuk industri telekomunikasi menggunakan *advance analysis* untuk memahami perilaku consumer dan hal tersebut dapat memprediksi asosiasi dari customer terhadap apakah customer tersebut akan meninggalkan layanan dari perusahaan. Dengan dilakukannya langkah prediktif tersebut, maka perusahaan telekomunikasi dapat menyusun langkah strategis terhadap segmentasi dari customer sehingga dapat di implementasikan strategi yang mengedepankan solusi terhadap permasalahan *customer churn* tersebut.

High Level Approach

Untuk menyelesaikan masalah ini, kami mempropose untuk pendekatan *predictive classification* menggunakan machine learning untuk menyelesaikan solusi permasalahan *churn customer*. Berdasarkan dataset yang diterima, terdapat fitur-fitur yang dapat digunakan untuk melakukan *predictive analysis* terhadap kemungkinan *churn customer*.

Dari dataset tersebut, akan dilakukan *preprocessing* untuk handling terhadap *missing value*, selanjutnya dilakukan pembagian dataset menjadi training, validation, dan testing set. Untuk fitur yang berbentuk kategori dilakukan One Hot Encoding dan Label Encoding untuk mengubah menjadi integer. Dilakukan juga Exploratory Data Analysis terhadap data training untuk mendapatkan gambaran terkait kondisi data. Tahap akhir dilakukan permodelan dengan menggunakan beberapa algoritma klasifikasi dimana pada data validation dilakukan Randomized Cross Validation untuk mendapatkan hyperparameter terbaik.

Setelah dilakukan modeling, dilakukan evaluasi terhadap performa model dan dilakukan deployment end-to-end. Dilakukan pytest untuk mengecek coding yang telah dibuat. Selanjutnya dilakukan deployment model menggunakan API dan Streamlit menggunakan Docker Container sebagai Virtual Environment agar service dapat digunakan oleh user untuk mendapatkan future data yang akan diprediksi. Deployment dapat dilakukan secara local dan secara online menggunakan AWS dan digunakan CICD untuk membuat Machine Learning Process secara otomatis.

Goal & Success

Metrics adalah “angka” atau “ukuran” yang menunjukkan informasi yang penting dari proses bisnis. Metrics ini berfungsi untuk memberikan informasi yang akurat sehingga dapat dijadikan dasar untuk perbaikan pada proses bisnis. Metrics yang baik adalah dapat dipahami, diingat, didiskusikan dan diinterpretasikan. Selain itu metric juga dapat dilakukan komparasi dari waktu-waktu,

Untuk mengukur performansi dari proyek, penulis membatasi permasalahan kepada performa model prediksi yang nantinya akan memfokuskan pada metrics F1 Score yang memperhatikan antara Precision

dan Recall. Dengan meningkatkan performa model tersebut dengan F1 Score yang tinggi, diharapkan didapatkan prediksi yang baik terhadap customer yang berpotensi churn yang nantinya akan berkaitan dengan bisnis *objective* yaitu memberikan meningkatkan tingkat retaining terhadap customer yang difokuskan churn tersebut dengan memberikan insentif, promosi, atau treatment lain yang nantinya akan berimpact terhadap Revenue perusahaan.

Key Solution

Untuk melakukan analisis prediktif terhadap klasifikasi, didapatkan dataset terhadap 3333 record terhadap customer telekomunikasi.

Adapun dataset yang digunakan adalah sebagai berikut:

- | | |
|--------------------|---|
| 1. ID | : Unique Classifier |
| 2. AccountWeeks | : Jumlah minggu customer yang memiliki akun yang aktif |
| 3. ContractRenewal | : - Renewal – customer melakukan renewal akun
- NotRenewal – customer tidak melakukan renewal akun |
| 4. DataPlan | : - 1 (Customer menggunakan data plan)
- 0 (Customer tidak menggunakan data plan) |
| 5. DataUsage | : Penggunaan data tiap bulannya (Gigabytes) |
| 6. CustSevCalls | : Jumlah telepon kepada <i>customer service</i> |
| 7. DayMins | : Rata-rata penggunaan per menit dalam sebulan |
| 8. DayCalls | : Rata-rata jumlah panggilan |
| 9. MonthlyCharge | : Rata-rata bulanan bill |
| 10. OverageFee | : Overage fee terbesar dalam 12 bulan |
| 11. RoamMins | : Rata-rata jumlah roaming dalam menit |

Data diambil dari *Kaggle* dan algoritma yang digunakan untuk klasifikasi adalah *Logistic Regression*, *KNN*, *Decision Tree*, *Random Forest*, dan *XGBoost* dimana pada data validation dilakukan Randomized Cross Validation untuk mendapatkan hyperparameter terbaik, hal tersebut digunakan untuk menghemat waktu dalam proses training.

Key Flows

Project pipeline yang digunakan dalam paper ini adalah menggunakan end-to-end machine learning hingga deployment model. Data structure dari project telah disesuaikan dengan *best practices* dalam pembuatan service machine learning dimana terdapat beberapa script python utama yang dijalankan yaitu:

- | | |
|--------------------------|---|
| 1. data_pipeline.py | : berisikan modul terkait data collection, data validation, data defence, data splitting |
| 2. preprocessing.py | : berisikan modul terkait handling missing values, One Hot Encoding, Label Encoding |
| 3. Modeling | : berisikan modul terkait baseline model, multimodel training, CV & Hyperparameter Tuning, Evaluation, dan Training Logging |
| 4. Pytest | : berisikan modul terkait pengetesan coding python |
| 5. Api.py & streamlit.py | : berisikan modul terkait API sebagai penghubung dengan user dan streamlit sebagai user interface |

Untuk service API dan streamlit ditunjang menggunakan Docker Service sebagai virtual environment, dan menggunakan AWS sebagai server dari layanan. Selanjutnya digunakan CICD untuk proses otomisasi program

Launch Readiness

Project dari timeline sesuai dengan deadline dari pengerjaan tugas proyek ML Process tanggal 04 Desember 2022.

Artifact

Komponen yang digunakan dalam proyek ini yaitu Visual Studio Code, Jupyter Notebook, WSL, Ubuntu, Google Chrome Browser, Git Hub, AWS EC2, Docker. Adapun modul Python yang digunakan adalah pandas, numpy, sklearn, matplotlib, seaborn, dan lainnya

References

A Survey on Customer Churn Prediction in Telecom Industry: Datasets, Methods and Metrics : V. Umayaparvathi¹, K. Iyakutti

<https://www.kaggle.com/datasets/becksddf/churn-in-telecoms-dataset/discussion/235073>

Source Code end to end workflow

1. Exploratory Data Analysis

Tahapan ini berada pada file Jupyter Notebook untuk melakukan analisis yang nantinya akan diubah kedalam bentuk *.py python file.

a. Cek ketersediaan *predictors*

File config.yaml tersedia untuk mempermudah dalam klasifikasi dari predictors, directory, range label yang nantinya akan digunakan dalam data pipeline. Adapun fitur yang digunakan sebagai predictor adalah AccountWeeks, ContractRenewal, DataPlan, DataUsage, CustServCalls, DayMins, DayCalls, MonthlyCharge, dan RoamMins. Sedangkan fitur Churn digunakan sebagai target

b. Cek data types

Saat pengecekan awal, data types untuk semua fitur merupakan object, sehingga dilakukan typecasting dataset ke object, dan integer sesuai dengan kebutuhannya

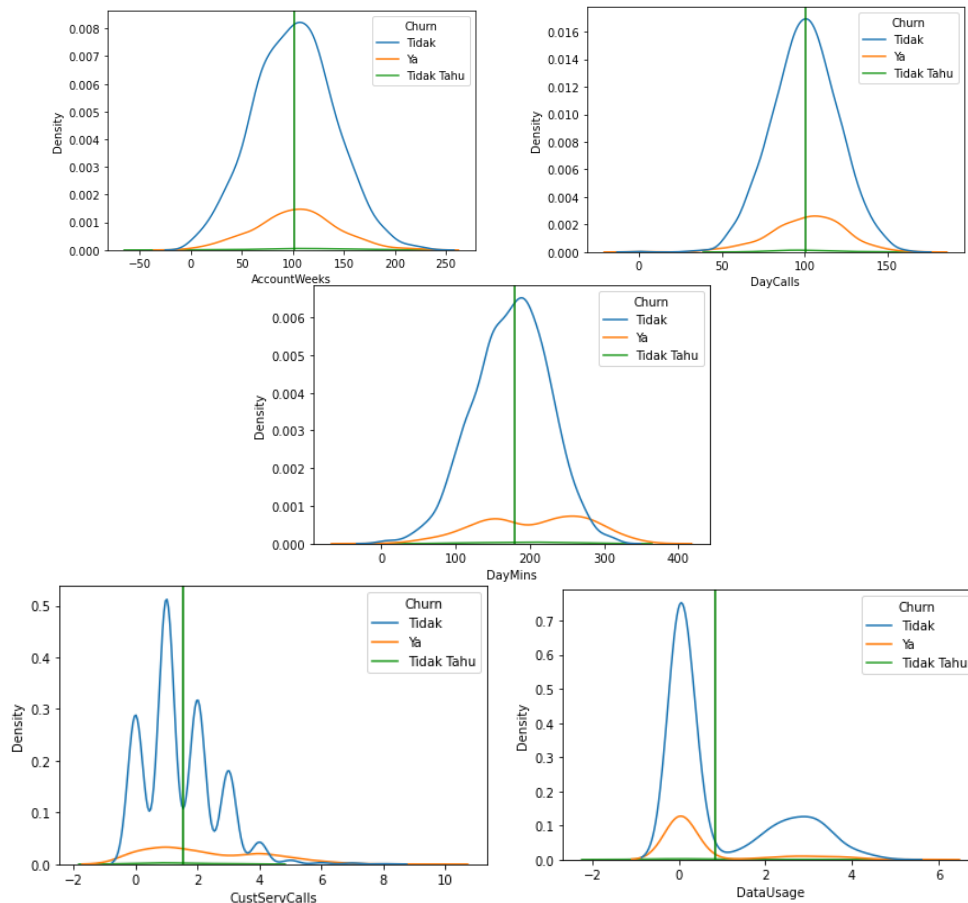
c. Cek data hilang, NaN, dan outliers

Dilakukan pengecekan terhadap data yang hilang dan didapatkan DataUsage : 5 data, DayMins : 11 data, OverageFee : 9 data. Selanjutnya terdapat data yang berupa "----" sehingga tidak dapat langsung dilakukan typecasting. Dilakukan perubahan terlebih dahulu ke -1 dan diubah Kembali menjadi NaN lalu dilakukan imputer

d. Cek range dari data

Pengecekan range data dilakukan untuk mendapatkan batas bawah dan batas atas dari data menggunakan boxplot dan melihat tingkat data yang imbalance.

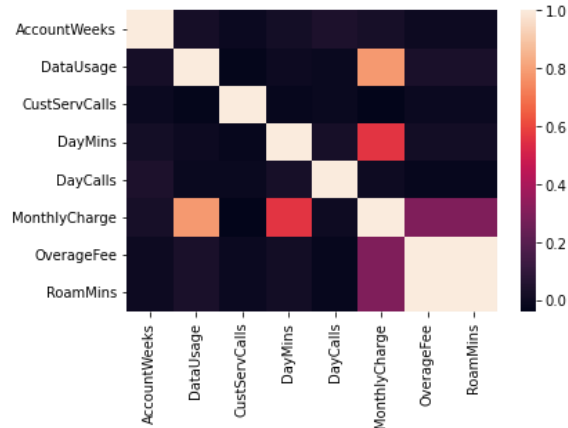
e. Pengecekan data imbalance atau skew



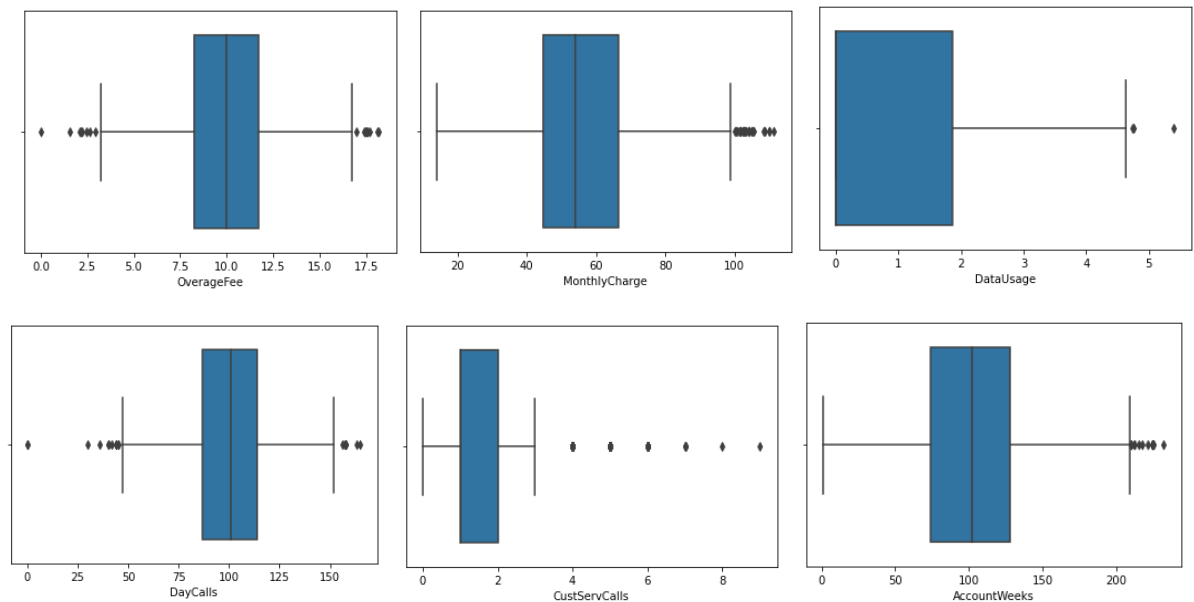
Berdasarkan plotting tersebut didapatkan nilai skewness yaitu AccountWeeks 0.081265, DataUsage 1.246628, CustServCalls 1.061323, DayMins 0.002133, DayCalls -0.148765, MonthlyCharge 0.550416, OverageFee -0.019028, RoamMins -0.019028. Dimana dapat dilihat bahwa AccountWeeks, DayMins, DayCalls, OverageFee, dan MonthlyCharge tidak memiliki skewness yang tinggi, sedangkan DataUsage dan CustServCalls memiliki skewness yang cukup tinggi

f. Pengecekan fitur lainnya

Berdasarkan hasil dari pengecekan dilakukan T-Test terhadap mean terhadap fitur yang churn dan tidak churn didapatkan gagal tolak H_0 sehingga dapat diasumsikan memiliki mean yang sama. Adapun penginputan median pada modul bertujuan untuk pembelajaran saja. Selanjutnya dengan pearson correlation dilakukan analisis terhadap hubungan antar fitur.

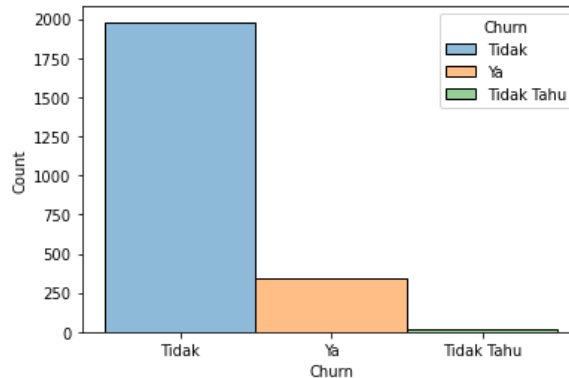


Dari fitur tersebut dapat dilihat bahwa MonthlyCharge berkorelasi positif dengan data usage, semakin banyak data usage yang digunakan maka tagihan akan semakin tinggi. Sementara untuk deteksi outlier digunakan boxplot sebagai berikut:



Berdasarkan boxplot, dapat dilihat bahwa rata-rata data terdistribusi normal dimana DataUsage memiliki skewness dan median pada nilai 1. Sedangkan CustServCalls memiliki

outliers. Karena kondisi data masih termasuk normal, maka tidak dilakukan handling outliers. Sedangkan dilihat dari jumlah data yang Churn dan Tidak Churn memiliki imbalance.



Untuk kategori “Tidak” terdapat 1980 observant, “Ya” terdapat 338 observant, dan “Tidak Tahu” sebanyak 15 orang. Data tersebut termasuk dalam kondisi imbalance sehingga untuk eksperimentasi akan digunakan Random Undersampling, Random Oversampling, dan SMOTE untuk melihat respon terhadap model.

2. Data Preprocessing

a. Defence and/or handling absence of predictors

Untuk handling terhadap ketersediaan predictor digunakan file config.yaml untuk mencocokkan fitur dari prediktor

b. Defence and/or handling wrong data type

Untuk handling terhadap type data yang salah, dilakukan type casting data dan dilakukan pencocokan terhadap isi data tersebut. Terdapat beberapa fitur yang memiliki nilai “----” sehingga dilakukan replace terlebih dahulu ke nilai -1.

c. Defence and/or handling missing data, nans, and outliers

Untuk missing data dan NaN, terdapat function nan_detector untuk mengecek fitur yang masih memiliki NaN value. Untuk fitur yang memiliki nilai -1 tadi di replace Kembali menjadi NaN dan dilakukan imputer sesuai dengan fiturnya

d. Defence and/or handling data out of range

Berdasarkan analisis hasil dari EDA, tidak terdapat data out of range yang perlu dilakukan handling

e. Defence and/or handling imbalance data or skewed data

Untuk data imbalance, dalam eksperimentasi digunakan 3 type sampling berbebeda untuk mendapat performa yang beragam. Untuk data yang skew pada predictor tidak dilakukan normalisasi.

3. Feature Engineering & Feature Selection

One Hot Encoding dilakukan terhadap fitur ContractRenewal yang bersifat kategorikal, sedangkan Label Encoder dilakukan terhadap fitur Churn agar berubah menjadi numerical. Feature Selection dilakukan dengan melakukan drop terhadap fitur “ID” yang merupakan unique identier yang mirip seperti label sehingga tidak memberikan efek terhadap model

4. Modeling

Pipeline dari modeling secara otomatis akan mengeluarkan record terkait logging terhadap training model yang berisi “model_name, model_uid, training_time, training_date, performance,

f1_score_avg, data_configurations". Baseline model yang digunakan adalah *Logistic Regression, Decision Tree, Random Forest, KNeighbors*, dan *XGBoost*. Selanjutnya, dimasukkan kedalam function *get_production_model* untuk mendapatkan model terbaik hasil produksi. Setelah itu, dilakukan training kembali terhadap data validation untuk mendapatkan Randomized Cross Validation sehingga didapatkan hyperparameter terbaik. Hasil dari model dapat di evaluasi menggunakan RandomForest dengan performa model sebagai berikut:

Kategori	Precision	Recall	f1-score	Support
0 (Tidak)	0.96	0.88	0.92	427
1 (Ya)	0.52	0.78	0.62	72
Accuracy	0.862			
Macro Avg	0.74	0.83	0.77	500
Weighted Avg	0.95	0.86	0.87	500

5. Serving

Sebelum dilakukan deployment, dilakukan debugging dan pengecekan secara otomatis menggunakan pytest sehingga model dapat berjalan dengan baik nantinya di API.

Setelah pipeline dari data_pipeline, preprocessing, dan modeling didapatkan dan dilakukan pengetesan, maka dilakukan serving deployment model. Terdapat dua service penting yang digunakan dalam deployment model yaitu API dan Streamlit. API digunakan sebagai jalur komunikasi antara Front End dan Back End sehingga data yang didapat dari user bisa digunakan untuk *predict data* menggunakan Fast API.

File dari api.py berisikan beberapa modul seperti FastAPI, Basemodel yang merupakan object class dari pydantic yang inherit dari Basemodel, uvicorn untuk web server yang digunakan oleh python, dan file python berupa pipeline yang telah dibuat. Setelah objek Fast API dibuat, maka dibuatkan struktur input data dan memuat model menggunakan class. Perlakuan yang diberikan pada data training model dilakukan juga terhadap *future* data dari API seperti Label Encoding dan One Hot Encoding.

Sedangkan untuk streamlit digunakan sebagai User Interface yang digunakan agar user dapat memasukkan data. File streamlit.py berisikan modul streamlit, request yang digunakan untuk melakukan request pada web yang mengizinkan untuk mengirimkan HTTP request, dan PIL image untuk memasukkan gambar. Selanjutnya dibuat number input dan select box menggunakan streamlit dengan range data yang sudah disesuaikan dengan hasil EDA dan dibuat submit button berupa "Predict". Hasil input tersebut dimasukkan kedalam JSON file dan dilakukan data defence terhadap potensi masalah yang muncul dan diberikan output akhir berupa prediksi churn.

Setelah dilakukan eksekusi pada terminal / wls.

Adapun cara untuk melakukan akses adalah dengan mengeksekusi pada terminal untuk file api dan menjalankan streamlit pada terminal secara terpisah.

```
PS C:\Users\Axel\Desktop\Data Science\Telecom Prediction\src> streamlit run .\streamlit.py
```

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8501>

Network URL: <http://192.168.18.11:8501>

```
PS C:\Users\Axel\Desktop\Data Science\Telecom Prediction\src> python .\api.py
INFO: Started server process [22352]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
```

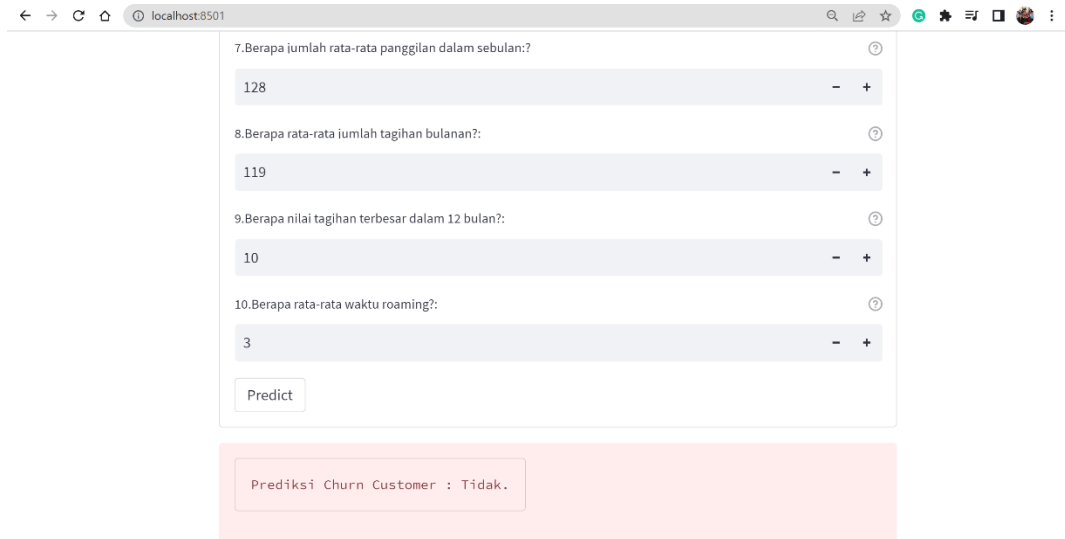
Berikut merupakan interface dari servis yang digunakan untuk prediksi terhadap customer churn pada perusahaan telekomunikasi



Prediksi Customer Churn pada Perusahaan Telecom

Isikan variable dibawah dah klik 'Predict':

1. Apakah customer melakukan perpanjangan kontrak masa berlaku?	?
Renewal	▼
2. Apakah customer menggunakan Data Plan untuk berkomunikasi? (Ya : 1 Tidak : 0):	?
0	- +
3. Total berapa minggu akun customer telah aktif?:	?
0	- +
4. Berapa total data usage perbulan yang digunakan customer (GB)?:	?
0	- +
5. Berapa total panggilan kepada customer service yang dilakukan?:	?
0	- +
6. Berapa menit rata-rata total penggunaan servis tiap bulan?:	?
0	- +
7. Berapa jumlah rata-rata panggilan dalam sebulan?:	?
0	- +
8. Berapa rata-rata jumlah tagihan bulanan?:	?
0	- +
9. Berapa nilai tagihan terbesar dalam 12 bulan?:	?
0	- +
10. Berapa rata-rata waktu roaming?:	?
0	- +
<button>Predict</button>	



7. Berapa jumlah rata-rata panggilan dalam sebulan?:
128

8. Berapa rata-rata jumlah tagihan bulanan?:
119

9. Berapa nilai tagihan terbesar dalam 12 bulan?:
10

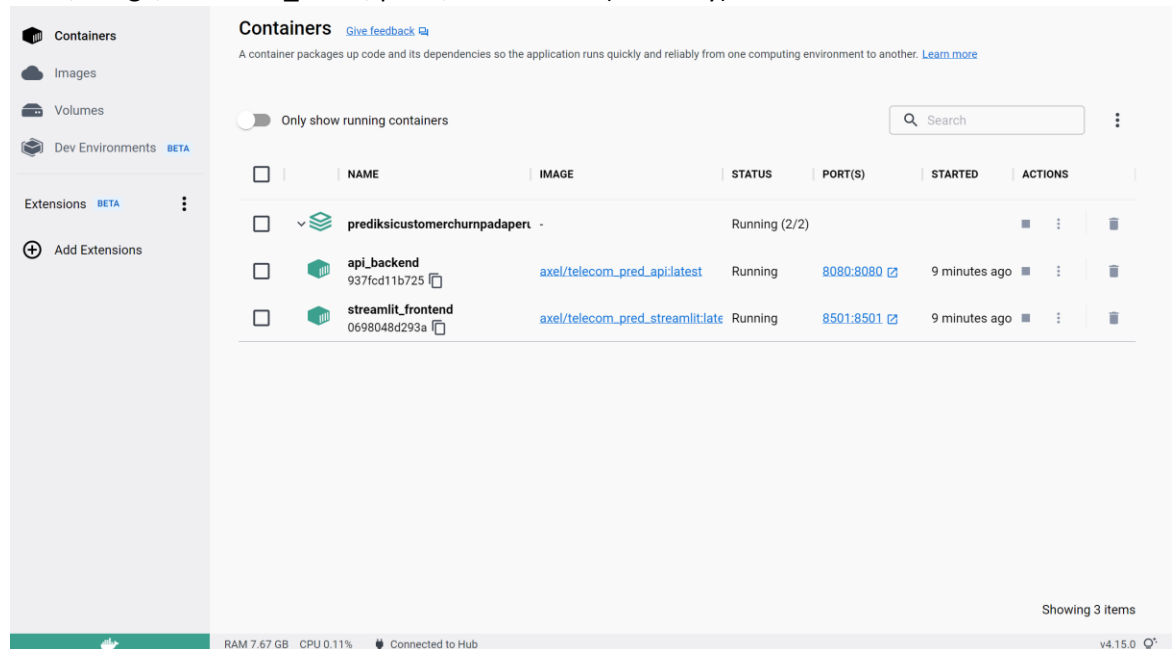
10. Berapa rata-rata waktu roaming?:
3

Predict

Prediksi Churn Customer : Tidak.

Berikut merupakan contoh hasil dari prediction model yang telah di deploy menggunakan Fast API menggunakan port 8080 dan Streamlit menggunakan port 8051. Setelah dilakukan penginputan terhadap setiap variable predictor, dan Klik Predict maka akan memprediksi nilai dari Churn Customer dimana hasil Prediksi Churn Customer tersebut adalah Tidak. Untuk kedua servis tersebut dijalankan secara bersamaan menggunakan Docker Container yang di mount untuk masing-masing servis menggunakan Dockerfile.

Pertama-tama docker di build terlebih dahulu sehingga menghasilkan docker image yang selanjutnya dilakukan run pada wsl/ubuntu. Dua buah docker file tersebut menggunakan python:3.9.15-slim-buster dan terdapat dua docker image untuk masing-masing servis. Docker akan melakukan install terhadap dependensi yang telah ditulis pada requirement.txt. Agar kedua docker dapat berjalan secara bersama, maka digunakan docker-compose.yml. File ini berisikan services yang digunakan (streamlit dan api), yang terdapat konfigurasi nama terhadap build, image, container_name, ports, dan volume (directory).



Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☐ Only show running containers

Search

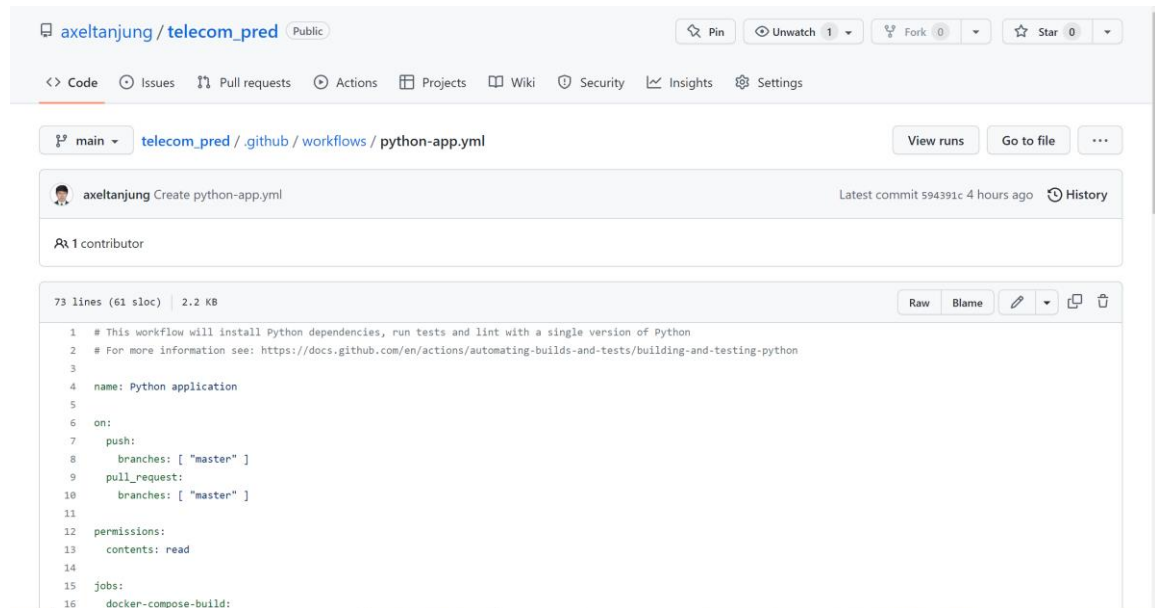
<input type="checkbox"/>	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
<input checked="" type="checkbox"/>	prediksicustomerchurnpadaperi		Running (2/2)			
<input type="checkbox"/>	api_backend 937fcd11b725	axel/telecom_pred_api:latest	Running	8080:8080	9 minutes ago	
<input type="checkbox"/>	streamlit_frontend 0698048d293a	axel/telecom_pred_streamlit:latest	Running	8501:8501	9 minutes ago	

Showing 3 items

RAM 7.67 GB CPU 0.11% Connected to Hub v4.15.0

Sampai tahapan ini, service sudah dapat berjalan pada localhost. Selanjutnya dilakukan deployment model secara online agar dapat diakses oleh user. Deployment dilakukan menggunakan server dari AWS menggunakan Instance pada EC2. Untuk menghubungkannya, seluruh file pipeline machine learning tersebut dimasukkan ke Git Hub. Untuk memasukkan ke Git Hub, dilakukan dengan membuat repository GitHub terlebih dahulu secara online.

Setelah repository dibuat, maka dilakukan “git init” untuk menginisiasi github pada folder working directory. Selanjutnya dilakukan “git add .” untuk track perubahan dokumen yang nantinya akan dilakukan “git commit”. Setelah dilakukan commit maka seluruh pipeline project dapat di push pada github. Selanjutnya untuk membuat deployment yang otomatis dapat digunakan CICD menggunakan github action dengan melakukan setting pada git-hub action



The screenshot shows the GitHub interface for the repository 'axeltanjung/telecom_pred'. The file path is '.github/workflows/python-app.yml'. The file is 73 lines (61 sloc) and 2.2 KB. The workflow is named 'Python application' and is triggered on 'push' to the 'master' branch. The workflow includes a 'permissions' section with 'contents: read' and a 'jobs' section with a job named 'docker-compose-build'.

```
1 # This workflow will install Python dependencies, run tests and lint with a single version of Python
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4 name: Python application
5
6 on:
7   push:
8     branches: [ "master" ]
9   pull_request:
10    branches: [ "master" ]
11
12 permissions:
13   contents: read
14
15 jobs:
16   docker-compose-build:
```