Syntax Literate: Jurnal Ilmiah Indonesia p–ISSN: 2541-0849 e-ISSN: 2548-1398
Vol. 7, No. 3, Maret 2022

# ANALISIS MASALAH, SOLUSI, DAN IMPLEMENTASI INTEGRASI MODEL VERIFIKASI KUALITAS GAMBAR DAN MODEL KLASIFIKASI ULOS TERHADAP APLIKASI DITENUN

**Arlinta Christy Barus, Johannes Harungguan Sianipar, Nico Gorga Soros Panjaitan, Ruthya P D Aruan, Reynaldo Hutahaean, Carina Lingga**
Fakultas Informatika dan Teknik Elektro, Institut Teknologi Del, Indonesia
Email: arlinta@del.ac.id@students.del.ac.id, johannes@del.ac.id,
ifs16041@students.del.ac.id, ifs16056@students.del.ac.id,
ifs16058@students.del.ac.id, ifs16059@students.del.ac.id

## Abstrak

Di Tenun merupakan sebuah aplikasi yang bertujuan untuk meningkatkan nilai jual kain tenun terutama Ulos dengan menciptakan motif baru. Saat ini aplikasi mobile Android DiTenun memiliki 3 fungsi utama yaitu generate kristik, generate motif dan ulos editor. Selain dari itu aplikasi DiTenun memiliki dua model machine learning yaitu, model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos. Mengintegrasikan kedua model ini dengan aplikasi DiTenun menjadi penting karena akan mempercanggih aplikasi DiTenun. Namun percobaan integrasi yang dilakukan oleh peneliti sebelumnya gagal. Dengan demikian penelitian ini bertujuan untuk mempelajari masalah apa yang dialami oleh peneli sebelumnya, mencari solusinya, dan mengimplementasikan model machine learning ke dalam aplikasi DiTenun. Penelitian ini menggunakan aplikasi mobile Android DiTenun dan model machine learning yang telah ada. Dalam penelitian ini integrasi model dengan aplikasi dilakukan dengan menginferensi model pada server DiTenun dan menyediakan layanan prediksi saat dibutuhkan oleh aplikasi melalui API. Untuk membangun API yang menyediakan layanan model digunakan dua framework yaitu Flask dan Django. Flask dan Django tidak hanya menerima permintaan dan memberi respon pada klien namun di dalamnya terdapat juga fungsi untuk mengubah gambar yang diterima menjadi bentukan yang sama dengan data testing pada model. Masing-masing model memiliki API Flask dan Djangonya sendiri yang menyediakan services yang terpisah. Berdasarkan pengujian yang telah dilakukan Flask dan Django sama-sama memberi kemudahan dalam mempelajari, struktur implementasi Flask lebih sederhana, sedangkan kecepatan respon aplikasi yang menggunakan Django lebih cepat daripada Flask. Dikarenakan pada penelitian ini kecepatan respon menjadi parameter penilaian utama, sedangkan yang lainnya adalah tambahan untuk saran, maka berdasarkan penelitian ini Django adalah solusi yang lebih baik untuk diterapkan.

**Kata Kunci**: aplikasi mobile ditenun model machine learning, model verifikasi kualitas gambar, model klasifikasi ulos, flask, django

**How to cite:** Arlinta Christy Barus. et al (2022) Analisis Masalah, Solusi, dan Implementasi Integrasi Model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos terhadap Aplikasi DiTenun. *Syntax Literate: Jurnal Ilmiah Indonesia, 7*(3).
**E-ISSN:** 2548-1398
**Published by:** Ridwan Institute

***Abstract***

*DiTenun is an application that aims to increase the selling value of woven fabrics, especially Ulos by creating new motifs. At the moment, the Android mobile application in DiTenun has 3 main functions namely, generate crystal, generate motif, and ulos editor. Aside from that DiTenun application has two machine learning models namely, the Image Quality Verification Model and the Ulos Classification Model. Integrating these two models with the DiTenun application is important because it will enhance the DiTenun application. However, integration experiments conducted by previous researchers failed. Thus this research aims to study what problems experienced by previous researchers, find solutions, and implement machine learning models into DiTenun applications. This research uses the Android application in DiTenun and existing machine learning models. In this research, the integration of the model with the application is done by interfering with the model on the DiTenun server and providing prediction services when needed by the application through the API. To build an API that provides model services two frameworks are used, namely Flask and Django. Not only does Flask and Django accept requests and respond to clients, but there are also functions to convert the received image to the same collision with the testing data on the model. Each model has its API Flask and Django which provide separate services. Based on testing that has been done Flask and Django both provide convenience in learning, the structure of Flask implementation is simpler, while the response speed of applications that use Django is faster than Flask. Because in this study the response speed is the main assessment parameter, while the others are additions to suggestions, based on this research Django is a better solution to be applied.*

***Keywords:*** *ditenun mobile application, machine learning model, image quality verification model, ulos classification model, flask, django*

Received: 2022-02-20; Accepted: 2022-02-05; Published: 2022-03-10

## Pendahuluan

Ulos merupakan kain tenun khas dari masyarakat Suku Batak yang menunjukkan kasih sayang antara orangtua dan anak, atau antara seseorang dengan orang lain (Aritonang, 2015). Selama bertahun-tahun motif Ulos yang sama diturunkan dari generasi ke generasi. Dengan demikian motif Ulos tersebut diproduksi secara terus menerus dengan minimnya perubahan yang terjadi. Untuk menghasilkan ide dan pasar yang lebih baik untuk kain Ulos maka dibutuhkan motif Ulos yang lebih beragam.

DiTenun adalah piranti lunak yang dikembangkan untuk menghasilkan motif baru dari pola tenun yang telah ada sebelumnya (Barus, Simanjuntak, & Situmorang, 2020). Aplikasi DiTenun memiliki beberapa modul, diantaranya modul Generate Motif, modul Generate Kristik dan modul Ulos Editor. Terdapat juga Model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos yang akan diintegrasikan dengan aplikasi DiTenun sebagai modul baru.

Model Verifikasi Kualitas Gambar adalah model *machine learning* yang dibangun untuk menyediakan layanan pengecekkan kualitas gambar Ulos baik atau buruk. Model ini dibangun menggunakan Algoritma *Support Vector Machine* (SVM) yang

diimplementasikan menggunakan bahasa pemrograman Python dan menggunakan *software* Jupyter Notebook. Model Klasifikasi Ulos adalah model *machine learning* menggunakan algoritma *Convolutional Neural Network* (CNN) untuk melakukan klasifikasi gambar Ulos berdasarkan pola Ulos. Model ini dibangun menggunakan bahasa pemrograman Python dan menggunakan *software* Jupyter dan Android Studio.

Model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos ingin diintegrasi ke aplikasi DiTenun sebagai sebuah modul baru, untuk menyediakan sebuah layanan agar aplikasi dapat menentukan apakah gambar yang dimasukkan berkualitas baik atau buruk. Kemudian setelah gambar dipisahkan berdasarkan klasifikasi baik atau buruk, gambar berkualitas baik dapat digunakan untuk menghasilkan kristik atau motif menggunakan Generate Kristik atau Generate Motif. Pengembang sebelumnya telah mencoba melakukan integrasi antara model Klasifikasi Ulos dengan aplikasi DiTenun dengan menggunakan API dan *web services*, namun proses integrasi tersebut gagal dilakukan dikarenakan *controller* tidak dapat memanggil file *Single Prediction* python.

Pada kebanyakan kasus, kegunaan *machine learning* pada sebuah aplikasi hanyalah bagian kecil dari aplikasi itu sendiri (Paul, 2020). Hal inilah yang terjadi dan membuat proses integrasi jadi kelihatan sulit. Ada dua hal yang biasanya dilakukan untuk menyelesaikan masalah integrasi tersebut:

1. Menulis ulang semua kode yang ditulis oleh pengembang. Hal ini kelihatan sebagai solusi yang baik, namun akan memakan banyak waktu dan energi, yang juga mungkin akan sia-sia. Kebanyakan bahasa pemrograman seperti Java Script tidak memiliki kemampuan yang cukup untuk melakukan *machine learning*.
2. *Web* API membuat bahasa pemrograman yang berbeda untuk saling bekerjasama dengan baik.

Oleh karena itu untuk menyelesaikan masalah integrasi antara Model Klasifikasi Ulos dan Model Verifikasi Kualitas Gambar terhadap aplikasi DiTenun maka peneliti memutuskan untuk membuat API sehingga kedua model tersebut bisa dikonsumsi oleh aplikasi android Di Tenun.

## Metode Penelitian

Metode penelitian yang akan dilakukan padad penelitian ini adalah sebagai berikut:

1. Studi Literatur

   Tahap ini bertujuan untuk pengumpulan bahan literatur dan informasi mengenai penelitian yang dilakukan. Dalam tahap ini dikaji hal-hal apa saja yang mempengaruhi integrasi model machine learning ke dalam aplikasi mobile berbasis android, jenis pengujian integrasi yang mungkin dilakukan dan penelitian terkait.
2. Analisis

   Tahap analisis dilakukan untuk menganalisis informasi yang diperoleh tentang proses integrasi yang sebelumnya terjadi pada aplikasi DiTenun dan menentukan masalah yang terjadi di dalam proses integrasi. Pada tahap ini juga peneliti melakukan uji coba menggunakan tools integrasi yang ditemukan.

3. Implementasi

   Pada tahap ini peneliti membuat rancangan integrasi dan mengimplementasikan metode integrasi yang dipilih pada tahap analisis. Pada tahap ini peneliti juga menentukan rancangan evaluasi tools dan rancangan pengujian kemudian melihat hasilnya.

4. Evaluasi

   Pada tahap ini merupakan tahapan untuk melakukan evaluasi terhadap proses penelitian yang telah dilakukan. Peneliti menentukan hasil dari proses integrasi yang telah dilakukan.

## Hasil dan Pembahasan

### Preparation

Pada tahap persiapan peneliti melakukan beberapa perubahan terhadap aplikasi mobile DiTenun, model Verifikasi Kualitas Gambar, dan model Klasifikasi Ulos. Berikut akan dijelaskan perubahan apa saja yang dikerjakan oleh penulis.

Pada aplikasi *mobile* DiTeun penulis menemukan bahwa terdapat beberapa *library* yang sudah usang (*obsolete*) pada proyek Android DiTenun. Artinya, terdapat beberapa *library* yang sudah tidak terdapat di *repository* yang di-*import* pada *file* build.gradle, sehingga proyek android tidak dapat di-*build*. Penulis kemudian melakukan konfigurasi pada *file* build. gradle, berdasarkan dengan *library* yang sesuai.

Terdapat juga masalah dalam pemanggilan API dikarenakan perpindahan server yang sebelumnya ke server DiTenun. Penulis melakukan perbaikan *bug* yang mengakibatkan aplikasi android tidak dapat mengakses server dan juga aplikasi yang tertutup secara tiba-tiba (*force close*). Penulis menambahkan *permission* pada proyek android untuk meng-*hit* yang sudah di-*deploy* di server DiTenun.

Secara lebih rinci, API untuk generate kristik dan generate motif sudah di-*deploy* di server, akan tetapi belum dikonfigurasikan pada aplikasi DiTenun, peneliti mengkonfigurasikan proyek android dalam meng-*hit* API yang telah di-*deploy* di server DiTenun. Selanjutnya peneliti mengatasi *error* yang disebabkan pengambilan data "motif saya" dari *database* yang statusnya masih kosong (*null*) saat aplikasi pertama kali di-install dalam perangkat (*device*) android.

Dalam percobaan ulang untuk mengintegrasikan model dengan aplikasi DiTenun peneliti menemukan bahwa beberapa *library* yang digunakan dalam pembangunan model ini tidak dapat digunakan lagi karena sudah usang (*obsolete*). Hal ini dikarenakan *library* Keras telah diperbaharui.

### Hasil Integrasi menggunakan Flask

Pada proses integrasi menggunakan Flask penulis membangun file Flask API yang berfungsi untuk menerima gambar dari aplikasi Android dan mengirim hasil klasifikasi. File Flask API dibangun menggunakan Bahasa Python. File ini juga berfungsi untuk mengubah gambar yang diterima menjadi *features* dan memanggil fungsi klasifikasi dari model Verifikasi.

Flask hanya membutuhkan satu *file* python dalam pengimplementasiannya (fungsi untuk *method* POST dan URL dalam satu *file* python berekstensi .py).

Satu *file* flaskklasulos.py sudah dapat meng-*handle* proyek Klasifikasi Ulos dengan Flask ini dan menjalankan proyek ini. Flask tidak memiliki aturan dalam pembuatan proyek, sehingga bergantung pada pengembang.

Jadi pada satu *file* python, kita sudah dapat me-*load* model, menyertakan juga URL dari fungsi yang tersedia, dan juga tidak membutuhkan *permission* untuk meng-*hit* API yang sudah ditentukan.

Setelah selesai membangun file Flask API penulis kemudian men*deploy* file di Heroku. Heroku adalah sebuah *cloud platform* atau tempat penyimpanan yang menjalankan bahasa pemrograman tertentu. Penulis kemudian menyimpan file Flask di git Heroku. Dengan demikian Heroku kemudian akan menjalankan file Flask Api di server yang dimiliki Heroku. Hasil verifikasi kemudian diterima oleh aplikasi Android kemudian ditampilkan.

Ketika Pengguna ingin memverifikasi kualitas gambar pengguna harus terlebih dahulu menambahkan gambar dengan cara memasukkan gambar melalui galeri atau menambahkan gambar melalui kamera. Setelah itu pengguna dapat menekan tombol ‘Verifikasi’ untuk mendapatkan hasil verifikasi. Berikut contoh hasil verifikasi yang dilakukan:

Proses integrasi model Klasifikasi Ulos menggunakan Flask sama dengan proses integrasi model Verifikasi. Perbedaannya hanya pada file Flask API yang dibangun untuk model Klasifikasi berbeda dengan Flask API untuk model Verifikasi.

**Hasil Integrasi Menggunakan Django**

Berbeda dengan Flask yang hanya membutuhkan satu file yang memanggil *library* Flask. Pada proses integrasi menggunakan Django penulis membangun proyek Django Rest API yang berfungsi untuk menerima gambar dari aplikasi Android dan mengirim hasil klasifikasi. Proyek Django API dibangun menggunakan Bahasa Python. Proyek ini juga berfungsi untuk mengubah gambar yang diterima menjadi *features* dan memanggil fungsi klasifikasi dari model Verifikasi.

Pada pengimplementasian Django, peneliti harus membuat proyek Django terlebih dahulu, mengikuti aturan Django dalam pembuatan suatu proyek. Jadi terdapat beberapa *folder* dan *file* yang saling berhubungan.

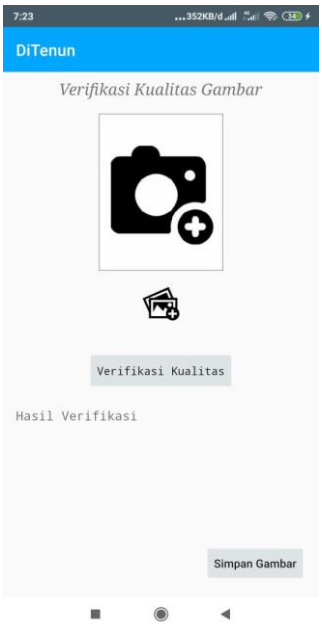
Dalam pengembangannya, peneliti melakukan perubahan terhadap beberapa file, diantaranya:

1. *file* view.py, dimana peneliti me-*load* model kemudian terdapat fungsi untuk meng-*handle method* POST.
2. *file* urls.py, dimana peneliti menentukan URL yang tersedia. Saat fungsi classify_api, yang ada di *file* view.py dipanggil, maka akan merujuk ke URL yang telah ditentukan.
3. *file* settings.py, dimana peneliti merubah ijin dari Klien yang dapat mengakses proyek ini menjadi *all host* (*)

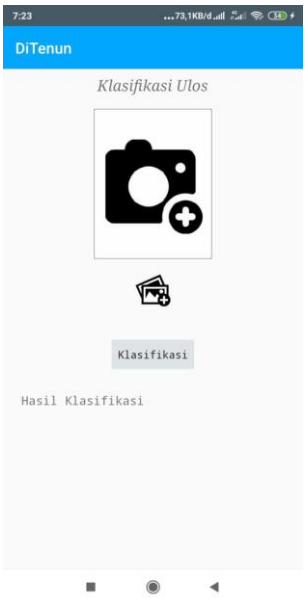
Setelah selesai membangun Proyek Django API penulis kemudian men*deploy* file di Heroku. Heroku adalah sebuah *cloud platform* atau tempat penyimpanan yang menjalankan bahasa pemrograman tertentu. Penulis kemudian menyimpan file Django di git Heroku. Dengan demikian Heroku kemudian akan menjalankan file Django Rest API di server yang dimiliki Heroku. Hasil verifikasi kemudian diterima oleh aplikasi Android kemudian ditampilkan.

**Hasil Integrasi dengan Aplikasi DiTenun**

Setelah memastikan API yang dibangun telah bekerja dengan baik maka proses selanjutnya adalah penulis mengintegrasikan model *machine learning* dengan aplikasi DiTenun saat ini. Hasil dari proses integrasi tersebut dapat dilihat pada gambar berikut:




**Gambar 1**
**Modul Verifikasi Kualitas Gambar**




**Gambar 2**
**Modul Klasifikasi Ulos**

Pada proses integrasi model *machine learning* dengan aplikasi DiTenun penulis meletakkan kedua model pada server DiTenun dan men*deploy* layanan melalui server DiTenun. Gambar di atas merupakan gambaran antarmuka kedua model pada aplikasi DiTenun menggunakan Flask dan Django.

Dalam men*deploy* kedua model *machine learning* dengan Flask dan Django, penulis masuk ke server DiTenun, kemudian memindahkan proyek Flask dan Django tersebut ke server. Selanjutnya, penulis membuat *environment* pada masing-masing metode dan model, sehingga terdapat empat *environment*. Penulis memiliki empat proyek, yaitu Klasifikasi Ulos dengan Django, Verifikasi Kualitas Gambar dengan Django, Klasifikasi Ulos dengan Flask, dan Verifikasi Kualitas Gambar dengan Flask. Kemudian, penulis menginstall *library* yang dibutuhkan pada setiap *environment*. Selanjutnya, penulis me-*running* keempat proyek itu pada server DiTenun pada *port* yang berbeda-beda.

Dibutuhkan *reverse proxy* untuk menerima *request* tertentu dari *klien* dan menghubungkannya ke server. Penulis memanipulasi URL agar bisa diarahkan ke masing-masing API yang telah di *deploy* di masing-masing port. Root URL server ditenun adalah http://mobile.ditenun.com. Contohnya, saat klien melakukan akses pada http://mobile.ditenun.com/flaskulos, maka *reverse proxy* akan mengarahkannya di dalam server ditenun ke http://localhost:8786/api/flasklas, dimana proyek Klasifikasi Ulos dengan Flask di *deploy*. Kemudian, penulis mengkonfigurasi *Endpoint* pada proyek android Ditenun sesuai dengan API yang telah tersedia.

**Hasil Pengujian**

Hasil pengujian unit yang dilakukan tidak menunjukkan adanya *error* oleh karena itu penulis kemudian melanjutkan ke integrasi model dengan Aplikasi DiTenun.

Berdasarkan hasil pengujian Integrasi didapatkanbahwa hasil Klasifikasi Ulos untuk model Klasifikasi tidak selalu tepat. Selain itu diperoleh hasil yang sama dari Flask dan Django untuk jenis *test case* yang sama. Berdasarkan hasil pengujian di atas dapat dilihat bahwa hasil Klasifikasi Ulos untuk model Klasifikasi tidak selalu tepat. Misalnya saat dimasukkan ulos Ragi Hotang hasil yang keluar adalah ulos Sibolang. Hal ini dikarenakan dalam penelitian sebelumnya persentase akurasi untuk model Klasifikasi adalah 54%. Dalam penelitian ini peneliti tidak melakukan perubahan untuk meningkatkan akurasi model. Oleh karena pada penelitian berikutnya integrasi model dengan mengikutsertakan peningkatan akurasi model Klasifikasi dapat dilakukan agar mendapatkan hasil yang lebih baik.

**Evaluasi Running Time Pada Server Heroku**

Setelah melakukan implementasi integrasi model *machine learning* dengan proyek Android yang dibangun serta men*deploy*nya ke server DiTenun maka penulis kemudian melakukan evaluasi *tools* integrasi yang digunakan. Pada proses melakukan evaluasi dengan mengukur kecepatan respon Flask dan Django.

Analisis yang digunakan adalah Paired Sample T-Test. Dimana output sig (2-tailed) atau p= 0,011. Berdasarkan uji Paired Sample T-Test, jika p< 0,05 maka terima Ho. Oleh karena itu terdapat perbedaan rata-rata antara integrasi menggunakan Flask dan Django. Dalam hal ini Flask lebih cepat daripada Django.

Dikarenakan hasil analisis dari Waktu Tunggu Hasil Klasifikasi menjadi parameter utama dalam mengambil keputusan maka berdasarkan kecepatan running time *tools* Flask lebih dipilih daripada Django.

**Evaluasi Tools**

**Evaluasi Kemudahan Mempelajari Tools**

Parameter ini merupakan parameter tambahan dalam mengevaluasi Flask dan Django. Parameter ini digunakan sebagai penilaian untuk saran bagi peneliti berikutnya dalam penelitian yang melibatkan Flask dan Django.

Flask dan Django merupakan *tools* yang baru pertama kali digunakan oleh penulis. Dengan kata lain penulis tidak memiliki pengetahuan sebelumnya terkait kedua *tools* ini. Dalam mempelajari Flask maupun Django penulis dimudahkan karena keduanya memiliki dokumentasi yang cukup lengkap.

Secara dokumentasi baik Django maupun Flask memiliki dokumentasi yang lengkap. Oleh karena itu keduanya dapat dikatakan seimbang dalam membantu *developer* untuk mempelajari *tools* masing-masing.

**Evaluasi Kompleksitas Implementasi Tools**

Parameter ini merupakan parameter tambahan dalam mengevaluasi Flask dan Django. Parameter ini digunakan sebagai penilaian untuk saran bagi peneliti berikutnya dalam penelitian yang melibatkan Flask dan Django.

Bagi seorang *newbie* atau pemula yang baru pertama kali mengenal dan mempelajari dua framework ini, Flask dan Django. Terdapat perbedaan yang mencolok dalam metode pengimplementasiannya. Dokumentasi kedua *framework* disajikan

dengan jelas dan dengan contoh, akan tetapi dalam pengimplementasiannya, Django membutuhkan lebih banyak aturan dan langkah yang lebih banyak dibandingkan Flask.

Dari penjelasan perbedaan pengimplementasian kedua metode. Hal ini bergantung daripada kebutuhan dan pengalaman dari *developer* .

Bagi *developer* yang masih baru atau awam dengan kedua metode, maka Flask menjadi metode yang cocok untuk membuat suatu web API. Flask memberikan kemudahan bagi penggunanya karena kesederhanaan dalam pengimplementasiannya.

Bagi *developer* yang sudah *expert*, maka sebaiknya disesuaikan dengan kebutuhan proyek dari *developer* tersebut. Dikarenakan Django sudah terbentuk dalam pattern MVC (Model-View-Controller), maka akan lebih memudahkan untuk proyek yang besar (*scalability*).

**Evaluasi Berdasarkan Running Time**

Parameter ini merupakan parameter utama dalam penelitian ini untuk mengevaluasi antara Flask dan Django. Hasil dari parameter ini menjadi pertimbangan utama dalam penelitian ini untuk membandingkan *tools* terbaik antara Flask dan Django.

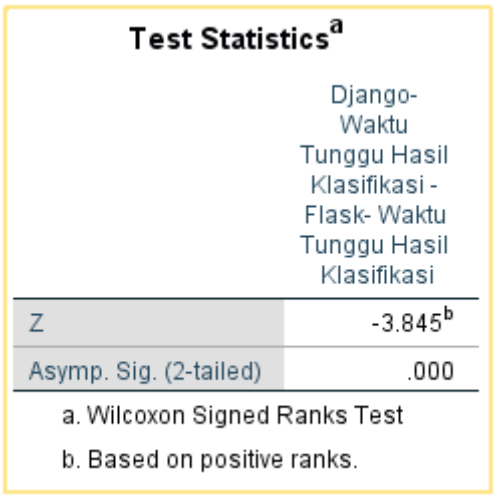
Pengujian dan pencatatan waktu pada server DiTenun dilakukan oleh 2 (dua) mahasiswa Kerja Praktek, yaitu Dian P.S Simanullang dan Diana Novita Sitio. Kedua mahasiswa tersebut diberikan *test case* yang sama dan melakukan pengujian mereka masing-masing.

Pada proses melakukan evaluasi dengan mengukur kecepatan respon Flask dan Django. Terdapat 2 (dua) variabel yang dihitung dan dicatat oleh penulis, yaitu:

1. Waktu Tunggu Hasil Klasifikasi: adalah waktu yang dibutuhkan aplikasi untuk menampilkan hasil klasifikasi. Dihitung sejak setelah ditekannya tombol ‘Verifikasi’ atau ‘Klasifikasi’ sampai hasil gambar keluar.
2. Total Waktu Run Test Case: adalah waktu yang dibutuhkan untuk menjalankan sebuah test case. Dihitung sejak proses membuka aplikasi, mengupload gambar sampai hasil klasifikasi keluar.

Dari kedua variabel di atas Waktu Tunggu Hasil Klasifikasi menjadi variabel utama dalam pemilihan antara Flask dan Django. Hal ini dikarenakan variabel ini menghitung secara spesifik kecepatan *tools* dalam memberi respon terhadap permintaan klien.

Berikut ini adalah hasil analisis untuk Waktu Tunggu hasil Klasifikasi:



**Test Statistics[a]**

| | Django-Waktu Tunggu Hasil Klasifikasi - Flask- Waktu Tunggu Hasil Klasifikasi |
|---|---|
| Z | -3.845[b] |
| Asymp. Sig. (2-tailed) | .000 |

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

**Gambar 3**
**Hasil Analisis Running Time**

Gambar di atas menampilkan hasil analisis Wilcoxon Signed Rank Test. Dimana output sig (2-tailed) atau p= 0,000. Berdasarkan uji Wilcoxon Signed Rank Test, jika p < 0,05 maka terima Ho. Oleh karena itu terdapat perbedaan rata-rata antara integrasi menggunakan Flask dan Django. Dalam hal ini Django lebih cepat daripada Flask.

Dikarenakan hasil analisis dari Waktu Tunggu Hasil Klasifikasi menjadi parameter utama dalam mengambil keputusan maka berdasarkan kecepatan running time *tools* Django lebih dipilih daripada Flask.

Berdasarkan hasil pada Heroku dan Flask terdapat perbedaan. Pada hasil analisis di Heroku, Flask lebih cepat daripada Django. Sedangkan berdasarkan analisis di server DiTenun, Django lebih cepat. Hal ini menunjukkan bahwa terdapat perbedaan hasil di keduanya. Namun, pada penelitian ini kebutuhan implementasi adalah di server DiTenun maka Django tetap dipilih.

## Kesimpulan

Berdasarkan penelitian yang telah dilakukan oleh penulis dapat diambil kesimpulan sebagai berikut: 1). Solusi yang dapat diterapkan dalam mengintegrasi model *machine learning* Verifikasi Kualitas Gambar dan model Klasifikasi Ulos ke dalam aplikasi mobile Android DiTenun adalah dengan cara menginferensi model ke dalam server DiTenun dan menyediakan layanan model ketika dibutuhkan aplikasi. Hal ini dapat dilakukan dengan cara membangun API menggunakan Flask dan Django. 2). Proses integrasi dilakukan dengan membangun API menggunakan Flask dan Django yang menerima sebuah gambar dan mengubah gambar tersebut menjadi bentuk yang sesuai dengan model. Kemudian baik Flask dan Django akan memanggil fungsi untuk memprediksi kelas gambar dan mengembalikan hasilnya ke aplikasi. 3). Model Verifikasi Kualitas Gambar dan Model Klasifikasi Ulos telah berhasil diintegrasikan dengan aplikasi *mobile* Android DiTenun. Masing-masing model yang diintegrasi dan metode integrasi memiliki *services* sendiri yang menyediakan layanannya. Hasil Model

Verifikasi Kualitas Gambar adalah sebuah layanan menambahkan gambar ke dalam aplikasi dan memeriksa apakah gambar yang ditambahkan memiliki kualitas yang baik atau buruk. Hasil Model Klasifikasi Ulos adalah sebuah layanan untuk memeriksa apakah sebuah gambar termasuk ke dalam 8 jenis ulos yang telah ditetapkan. 4). Berdasarkan evaluasi metode integrasi menggunakan running time *test case* hasil integrasi menggunakan Django lebih cepat daripada Flask pada server DiTenun. Oleh karena itu pada kasus penelitian ini Django lebih dipilih.

## BIBLIOGRAFI

Aritonang, Sondang Daniel. (2015). *Prospek Industri Tenun Ulos di Kabupaten Toba Samosir*. Riau University. Google Scholar

Barus, Arlinta C., Simanjuntak, Marianna, & Situmorang, Verawati. (2020). DiTenun, Smart Application Producing Ulos Motif. *SHS Web of Conferences*, *86*, 1025. EDP Sciences. Google Scholar

Paul, S. (2020). *Turning Machine Learning Models into APIs in Python. Diambil kembali dari DataCamp*. Retrieved from https://www.datacamp.com/community/tutorials/machine-learning-models-api-python