JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 11 No. 2, pISSN: 2303-0577 eISSN: 2830-7062 DOI: http://dx.doi.org/10.23960/jitet.v11i2.3009

RANCANG BANGUN APLIKASI WEB PENDETEKSI WARNA PADA *PIXEL* GAMBAR DENGAN *KNN CLASSIFIER*

Kadafi Eka Sakti [1*], Mardiana[2] , Rio Ariestia Pradipta[3]

[1,2,3] Teknik Informatika, Universitas Lampung; Jl. Prof. Dr. Ir. Sumantri Brojonegoro No.1, Bandar Lampung, Lampung

Riwayat artikel:
Received: 2 April 2023
Accepted: 10 April 2023
Published: 12 April 2023

Keywords:
K-Nearest Neighbor;
AI Project Cycle;
Flask;
Usability Testing.

Corespondent Email:
kadafieka@gmail.com

Abstrak. Manusia memiliki keterbatasan dalam mendeteksi ratusan warna dalam gambar secara presisi. Banyak orang awam mengalami kesulitan dalam mengklasifikasikan sebuah warna pada benda ,pada umumnya orang awam hanya mengetahui warna dasar atau warna yang umum. Sehingga aplikasi web bernama DYSCO ini dibuat untuk mempermudah dalam mengklasifikasikan sebuah warna. DYSCO merupakan aplikasi pendeteksi warna berbasis *web* dengan menggunakan metode *K-Nearest Neighbors*. Aplikasi web ini dibuat dengan menggunakan bahasa pemrograman *Python* dan kerangka kerja mikro *Flask.* Aplikasi web ini dapat menampilkan informasi warna *pixel* suatu gambar yang di-*upload* pengguna ke dalam *web*, dengan mengektraksi fitur *RGB* pada gambar tersebut. Metode pengembangan yang digunakan adalah *AI Project Cycle* dengan 6 tahapan yaitu *Problem Scoping, Data Acquisition, Data Exploration, Modelling* ,*Evaluation dan Deployment*. Pembuatan aplikasi web pendeteksi warna pada *pixel ini* telah berhasil dilakukan dan sudah mampu mendeteksi dan memberikan informasi warna yang di klik pengguna pada gambar, dengan akurasi yang tinggi yaitu 0,844. Aplikasi web juga sudah dapat memenuhi fungsi fungsi utamanya, ditunjukan melalui pengujian menggunakan teknik *Usability Testing* berdasarkan ISO 9421-11 untuk mengukur *Efectivity, Eficiency* dan *Satisfaction*, dengan 10 orang responden awam. Hasil *Usability Testing* mendapatkan skor yang sangat bagus dengan nilai *efectivity* yaitu 100% dengan predikat sangat efektif, nilai *eficiency* yaitu 32,6 detik dengan predikat sangat cepat, dan nilai *satisfaction* yaitu 84 dengan predikat dapat diterima (*acceptable*).

Abstract. *Humans have limitation in detecting the hundreds of colors in an image with precision. Many ordinary people have difficulty classifying a color in objects, in general, ordinary people only know basic colors or common colors. So a web application called DYSCO was made to make it easier to classify a color. DYSCO is a web-based color detection application using the K-Nearest Neighbors method. This web application is built using the Python programming language and the Flask micro framework. This web application can display pixel color information of an image that the user uploads to the web, by extracting the RGB features of the image. The development method used is the AI Project Cycle with 6 stages, namely Problem Scoping, Data Acquisition, Data Exploration, Modeling, Evaluation and Deployment. The creation of this pixel color detection web application has been successfully carried out and has been able to detect and provide color information that user clicks on the image, with a high accuracy of 0.844. The web application*

has also been able to fulfill its main functions, shown through testing using the Usability Testing technique based on ISO 9421-11 to measure Effectiveness, Efficiency and Satisfaction, with 10 respondents. Usability Testing results get a very good score with an effectiveness value of 100% with a very effective predicate, an efficiency score of 32.6 seconds with a very fast predicate, and a satisfaction score of 84 with an acceptable predicate.

1. PENDAHULUAN

Klasifikasi sebuah gambar dalam suatu label tertentu adalah salah satu hal yang sangat mudah bagi manusia karena pada hakikatnya manusia merupakan makhluk visual yang bisa melihat, mendeteksi serta mengklasifikasi bentuk data baik data berupa teks ataupun data gambar yang dikolektifkan menjadi sebuah dataset yang dapat dianalisis. Namun, pada dasarnya manusia juga memiliki keterbatasan dalam mengklasifikasi suatu hal. Pada penelitian akhir ini, dengan membawa latar belakang masalah klasifikasi warna berdasarkan *pixel* gambar, hal ini dikarenakan manusia memiliki keterbatasan dalam mengklasifikasi dan mendeteksi warna warna yang ada pada gambar. *Pixel* itu sendiri berunsur dari 3 warna yaitu hijau, biru, dan merah (RGB) dan dari setiap *inch* warna itu dapat menghasilkan 16 juta warna dari perubahan atau penambahan *inch* tersebut . Oleh karena itu, solusi dari permasalahan ini adalah menggunakan bantuan metode AI yaitu *K-Nearest Neighbor* (K-NN) dalam mengidentifikasi warna-warna tersebut menjadi suatu label nama warna yang presisi.

Dalam penelitian ini dilakukan pembuatan sebuah aplikasi web dengan kode nama DYSCO yang dapat digunakan untuk mengidentifikasi warna dari gambar. Tujuan pembuatan aplikasi web ini adalah untuk membantu dalam kesulitan pengenalan warna dalam sebuah gambar yang dapat dialami oleh pengguna, pengguna disini adalah seseorang yang mengidap buta warna ataupun orang awam yang ingin tahu suatu warna dari sebuah gambar. Pengguna dapat mengunggah gambar ke situs web dan mengklasifikasi warna dalam gambar sesuai keinginan Pengguna dengan cara mengeklik kursor pada warna dalam gambar untuk mengklasifikasikan dan mendeteksi warna dari sebuah gambar. Informasi warna yang didapat dapat digunakan pengguna untuk berbagai hal , bagi pengguna pengidap buta warna informasi ini dapat digunakan untuk membandingkan dengan warna lain, untuk pengguna lain informasi ini dapat digunakan untuk berbagai bidang seperti desain yang membutuhkan informasi akurat mengenai warna. Ruang warna utama yang digunakan dalam penelitian ini adalah RGB (*Red, Green, Blue*). Aplikasi web dibangun menggunakan *Flask.* Implementasi dari penelitian ini adalah dengan menggunakan algoritma K-NN, KNN dipilih sebagai algoritma dikarenakan berdasarkan beberapa penelitian disimpulkan bahwa KNN dapat digunakan dalam klasifikasi warna dengan tingkat akurasi yang tinggi lebih dari 90% (0,9), KNN dilatih dengan nilai histogram warna R, G, dan B, warna dari citra masukan yang diberikan akan dideteksi dan kemudian dievaluasi modelnya serta dapat menghasilkan output berupa nama warna sesuai dengan metode klasifikasi K-NN.

2. TINJAUAN PUSTAKA

2.1. *Flask*

Flask adalah kerangka kerja mikro pengembangan web yang ditulis dengan *Python*. Mudah dipelajari dan digunakan. *Flask* "ramah pemula" karena tidak ada kode *boilerplate* atau dependensi yang mengalihkan perhatian dari fungsionalitas inti aplikasi, Kerangka kerja mikro berbeda dari kerangka kerja *full-stack* yang juga menyediakan modul tambahan untuk fitur seperti otentikasi, ORM basis data, validasi input, sanitasi, dll.

Flask dikenal sebagai *micro-framework* karena ringan dan hanya berisi komponen yang diperlukan. *Flask* hanya menyediakan komponen yang dibutuhkan untuk pengembangan web, seperti: *routing*, pemrosesan *request*, sesi, dll. Untuk fungsi lain seperti pemrosesan data, pengembang dapat menulis modul mereka sendiri atau menggunakan ekstensi. Pendekatan ini menghindari kode

boilerplate yang tidak perlu yang sebenarnya tidak digunakan [3].

Berikut adalah beberapa fitur yang menjadikan *Flask* sebagai *framework* ideal untuk pengembangan aplikasi web.

1. *Flask* menyediakan *server* pengembangan dan *debugger*.
2. Menggunakan *template Jinja2*. *3*. Mendukung WSGI 1.0.
3. Dukungan bawaan untuk pengujian unit.
4. *Flask* menyediakan banyak ekstensi yang dapat digunakan untuk memperluas fungsionalitasnya.

2.2. *K-Nearest Neighbor*

KNN (*K-Nearest Neighbor*) adalah algoritma pembelajaran AI, algoritma ini termasuk dalam *supervised learning*. Algoritma ini mengasumsikan bahwa tiap *instance* dari data adalah bertetangga dan *instance* yang berdekatan dianggap memiliki kelas yang sama, tujuan dari algoritma ini adalah untuk mengklasifikasikan objek baru berdasarkan atribut dan sampel dari *training* data [4].

Algoritma KNN menggunakan *Neighborhood Classification* sebagai nilai prediksi dari nilai *isntance* yang baru. Secara umum alur kerja K-NN adalah sebagai berikut :

1. Menentukan jumlah tetangga (K) yang akan digunakan untuk pertimbangan penentuan kelas
2. Menghitung jarak dari data baru ke masing masing data pada *dataset* untuk menentukan tetangga terdekat
3. Menentukan nilai jarak terdekat dan penentuan kelas.
4. Klasisfikasi dengan mengambil nilai K, kemudian menentukan kelas dari data baru

Rumus yang digunakan untuk menetukan jarak adalah dengan perhitungan *Euclidean Distance* seperti berikut :

$$dis = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2 + (y_{1i} - y_{2i})^2 + \cdots}$$

Gambar 1. Rumus Euclidean Distance

Pada Gambar 2 diatas x_1 adalah data yang akan di prediksi dan adalah x_2 data pada *entry* pertama. Dalam prakteknya di *python* dapat digunakan kode program untuk mendapatkan nilai K terbaik dari berbagai opsi [2].

2.3. *Usability Testing*

Usability Testing merupakan proses belajar tentang pengguna dari pengguna dengan mengamati mereka menggunakan produk untuk mencapai tujuan tertentu .

Menurut ISO 9241-11, ada tiga aspek yang dijadikan acuan untuk mengukur seberapa bergunanya aplikasi dalam membantu mencapai tujuan tertentu oleh pengguna aplikasi. Ketiga aspek usability tersebut, yaitu: *Effectiveness* (kemampuan pengguna untuk mencapai tujuan mereka dalam konteks yang spesifik), *Efficiency* (sumber daya yang digunakan terkait dengan akurasi dan kesempurnaan yang dicapai pengguna dalam menjalankan tugas), dan *Satisfaction* (tingkat kepuasan pengguna saat menggunakan aplikasi*). Usability Testing* adalah pengujian penggunaan terhadap sistem atau produk untuk menemukan permasalah daya guna atau *usability* . Dalam pengujian *usability* terdapat satu teknik yang bisa digunakan dalam mengukur tingkat efektifitas dan efisiensi suatu produk, yaitu: teknik *performance measurement* . Teknik tersebut dapat memberikan hasil pengukuran yang lebih akurat dikarenakan sistem pengujiannya langsung berdasarkan pengalaman penggunanya. Di sisi lain, *System Usability Scale* (SUS) dapat memberikan pandangan subjektif tentang usability dari suatu sistem . Nilai yang didapatkan dari metode SUS dapat dijadikan pertimbangan tingkat kelayakan suatu aplikasi untuk diterapkan . Oleh karena itu, kuesioner SUS dapat digunakan untuk mengukur tingkat kepuasan pengguna terhadap suatu produk karena sifat penilaiannya yang subjektif .

2.3.1. Teknik Performance Measuement

Untuk melihat kefektifan akan dilihat dari jumlah *task* yang berhasil diselesaikan responden. Penyelesaian dihitung dengan menetapkan angka biner ‘1’ jika partisipan berhasil dan ‘0’ jika partisipan gagal. Rumus yang digunakan adalah sebagai berikut:

$$Efektifitas = \frac{\text{Jumlah tugas yang berhasil diselesaikan}}{\text{Jumlah total tugas}} x100\%$$

Gambar 2. Rumus Penilaian Efektifitas

Pada Gambar 2 diatas dijabarkan untuk memperoleh hasil efektifitas, banyaknya tugas yang diselesaikan dan berhasil dibagi banyaknya total tugas yang diberikan yang hasilnya dikalikan 100%. Rata-rata penyelesaian tugas minimum pada pengujian *usability* adalah 78%, namun bila hasil dibawah 49% menempatkan pada kuartil bawah .

Efektivitas harus dinilai atas dasar tujuan yang bisa dilaksanakan, bukan atas dasar konsep tujuan yang maksimum. Efektivitas diukur dengan menggunakan standar sesuai dengan acuan Litbang Depdagri (1991) seperti pada Tabel 1 [5].

Tabel 1. Standar Ukuran Efektifitas

Rasio Efektivitas	Tingkat Capaian
Dibawah 40	Sangat Tidak Efektif
40 - 59.99	Tidak Efektif
60 – 79.99	Cukup Efektif
Diatas 80	Sangat Efektif

Sedangkan untuk mengukur Keefisiensian dapat diukur dalam waktu tugas, yaitu waktu dalam hitungan menit/detik yang dibutuhkan untuk menyelesaikan tugas dengan sukses. Rumus yang digunakan untuk menghitung keefisienan suatu produk atau layanan adalah sebagai berikut: Waktu yang diperoleh diukur dengan mengurangkan waktu selesai dengan waktu mulai, kemudian dihitung menggunakan persamaan pada Gambar 3 berikut.

$$\text{Efisiensi efektif keseluruhan} = \frac{\sum_{j=1}^{R}\sum_{i=1}^{N}\frac{n_{ij}}{t_{ij}}}{NR}$$

Gambar 3. Rumus Perhitungan Efisiensi

Keterangan :
N = Jumlah total tugas (gol)
R = Jumlah pengguna
nij = Hasil tugas i oleh pengguna j; jika pengguna berhasil menyelesaikan tugas, maka Nij = 1, jika tidak, maka Nij = 0 ,tij = Waktu yang dihabiskan oleh pengguna j untuk menyelesaikan tugas i. Jika tugas tidak berhasil diselesaikan, maka waktu diukur hingga saat pengguna berhenti dari tugas .

Selanjutnya hasil rata-rata waktu yang dibutuhkan untuk mengerjakan *task scenario* tersebut diinterpretasikan menggunakan *range* waktu pada indikator *time behavior* seperti yang terdapat pada Tabel 2 untuk menentukan durasi waktu yang digunakan pengguna dalam penyelesaiaan task sehingga dapat diketahui hasil pengukuran tingkat efisiensi dari aplikasi [5].

Tabel 2. Standar Ukuran Efisiensi

No	Lamanya Waktu	Kualifikasi
1	60 -300 *Second*	Sangat Cepat
2	360 - 600 *Second*	Cepat
3	660 - 900 *Second*	Lambat

2.3.2. System Usability Scale

Salah satunya pengujian *usability* bisa dilakukan menggunakan *System Usability Scale* (SUS) . SUS memiliki 10 pertanyaan dan 5 pilihan jawaban. Pilihan jawaban terdiri dari sangat tidak setuju sampai sangat setuju. SUS memiliki skor minimal 0 dan skor maksimal 100. SUS dalam bahasa aslinya menggunakan bahasa Inggris. Berikut 10 pertanyaan dari *System Usability Scale* (SUS) yang sudah diterjemahkan dalam bahasa Indonesia [6]:

No.	Item in Indonesian
1	Saya berpikir akan menggunakan sistem ini lagi.
2	Saya merasa sistem ini rumit untuk digunakan.
3	Saya merasa sistem ini mudah untuk digunakan.
4	Saya membutuhkan bantuan dari orang lain atau teknisi dalam menggunakan sistem ini.
5	Saya merasa fitur-fitur sistem ini berjalan dengan semestinya.
6	Saya merasa ada banyak hal yang tidak konsisten (tidak serasi) pada sistem ini.
7	Saya merasa orang lain akan memahami cara menggunakan sistem ini dengan cepat.
8	Saya merasa sistem ini membingungkan.
9	Saya merasa tidak ada hambatan dalam menggunakan sistem ini.
10	Saya perlu membiasakan diri terlebih dahulu sebelum menggunakan sistem ini.

Gambar 4. Pertanyaan SUS

Gambar 4 diatas menampilkan daftar 10 pertanyaan SUS. SUS memiliki 5 pilihan jawaban. Mulai dari sangat tidak setuju, tidak setuju, ragu-ragu, setuju, dan sangat setuju. Skor masing-masing jawaban mulai dari 1 sampai 5. Berikut pilihan jawaban beserta skornya.

Jawaban	Skor
Sangat Tidak Setuju (STS)	1
Tidak Setuju (TS)	2
Ragu-ragu (RG)	3
Setuju (S)	4
Sangat Setuju (SS)	5

	STS	TS	RG	ST	SS
1. Saya berpikir akan menggunakan sistem ini lagi.					
	1	2	3	4	5

Gambar 5. *Chart* Nilai Pertanyaan SUS

Gambar 5 adalah sebuah *chart* penilaian setiap jawaban SUS. Setelah melakukan pengumpulan data dari responden, kemudian data tersebut dihitung. Dalam cara menggunakan *System Usability Scale* (SUS) ada beberapa aturan dalam perhitungan skor SUS. Berikut ini aturan-aturan saat perhitungan skor pada kuesionernya:

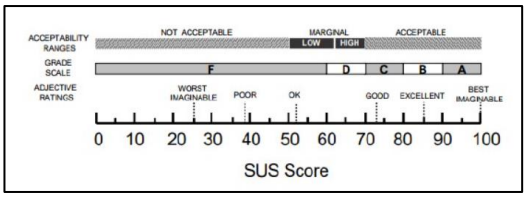
a. Setiap pertanyaan bernomor ganjil, skor setiap pertanyaan yang didapat dari skor pengguna akan dikurangi 1.
b. Setiap pertanyaan bernomor genap, skor akhir didapat dari nilai 5 dikurangi skor pertanyaan yang didapat dari pengguna.
c. Skor SUS didapat dari hasil penjumlahan skor setiap pertanyaan yang kemudian dikali 2,5.

$$\bar{x} = \frac{\sum x}{n}$$

$\bar{x}$ = skor rata-rata
$\sum x$ = jumlah skor SUS
n = jumlah responden

Gambar 6. Rumus Perhitungan SUS

Gambar 6 menunjukkan rumus perhitungan SUS yaitu adalah rata rata dari seluruh jumlah skor sus dibagi dengan jumlah responden. Aturan perhitungan skor untuk berlaku pada 1 responden. Untuk perhitungan selanjutnya, skor SUS dari masing-masing responden dicari skor rata-ratanya dengan menjumlahkan semua skor dan dibagi dengan jumlah responden.



Gambar 7. Grafik Kesimpulan SUS

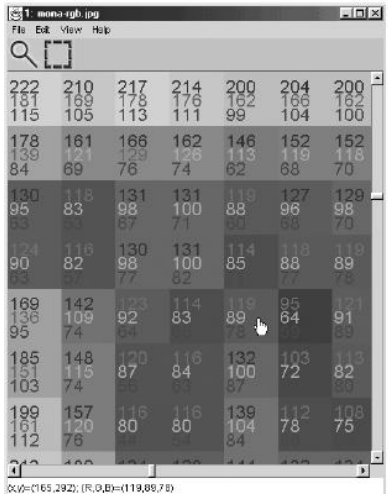
Gambar 7 adalah grafik kesimpulan nilai skor SUS ada 3 teknik penilaian yaitu berdasarkan *acceptability range, grade scale* dan *adjective ratings*. Kesimpulan dari cara menggunakan *System Usability Scale* (SUS) adalah setelah dihitung didapatlah skor rata-rata SUS dari semua responden. Skor tersebut kemudian disesuaikan dengan penilaian SUS. Masuk kategori mana hasil pengujian dengan skor rata-rata yang sudah didapat. Skor rata-rata SUS dari banyaknya penelitian adalah 68, maka jika nilai SUS di atas 68 akan dianggap di atas rata-rata dan nilai di bawah 68 di bawah rata-rata. Jika skor yang kamu dapat dibawah 68 berarti ada masalah pada *usability* dan butuh perbaikan. Namun kesimpulan akhir bisa juga ditentukan melalui penilaian seperti pada gambar diatas [6].

2.4 RGB

Untuk melihat kefektifan akan dilihat dari jumlah *task* yang berhasil diselesaikan responden. Penyelesaian dihitung dengan menetapkan angka biner '1' jika partisipan berhasil dan '0' jika partisipan gagal. Rumus yang digunakan adalah sebagai berikut: Model warna RGB (*Red, Green, Blue*) adalah model warna tambahan di mana warna primer cahaya merah, hijau dan biru ditambahkan bersama-sama dalam berbagai cara untuk mereproduksi susunan warna yang luas. Nama model berasal dari inisial dari tiga warna primer aditif, merah, hijau, dan biru .

Tujuan utama dari model warna RGB adalah untuk penginderaan, representasi, dan tampilan gambar dalam sistem elektronik, seperti televisi dan komputer, meskipun juga telah digunakan dalam fotografi konvensional. Sebelum zaman elektronik, model warna RGB sudah memiliki teori yang kuat di belakangnya, berdasarkan persepsi warna manusia.

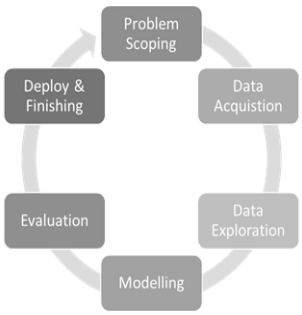
Perbesar terus sebuah gambar dan akan segera berada pada skala di mana masing-masing piksel akan terlihat. Setiap piksel adalah wilayah persegi kecil dari gambar. Warnanya diwakili oleh tiga angka: komponen merah, komponen hijau, dan komponen biru. ketiga komponen piksel ini dikenalsebagai nilai RGB piksel atau RGB *triple*. Nilai terkecil yang dapat muncul dalam RGB *triple* adalah 0. Misalnya RGB *triple* (0, 0, 0) mewakili warna yang memiliki 0 satuan merah, 0 satuan hijau, dan 0 satuan biru, warna ini adalah hitam., nilai terbesar yang dapat muncul dalam triple RGB adalah 255, dan RGB *triple* (255, 255, 255) mewakili warna putih [7].



Gambar 8. Representasi RGB

Pada Gambar 8 menunjukkan nilai setiap piksel dan nilai RGB pada koordinat yang juga merepresentasikan intensitas warna pada citra berwarna yang dapat dikodekan dalam 24 bit, masing masing nilai RGB dikodekan dengan 8 bit, dengan demikian sebuah citra berwarna dapat memiliki kandungan warna maksimum sebanyak 2^{24} atau sebanyak 16.777.216 variasi warna .

3. METODE PENELITIAN



Gambar 9. Alur AI Project Cycle

Penelitian dilakukan menggunakan alur AI *Project Life Cycle* atau Siklus hidup proyek AI,Siklus hidup proyek AI adalah fase atau metode kerja untuk membuat proyek AI. Siklus hidup proyek AI dapat dibagi menjadi enam fase seperti yang ditunjukkan pada diagram pada Gambar 1 dengan penjelasan sebagai berikut [1] :

1. *Problem Scoping*

Pada fase ini, diidentifikasi masalah yang akan diselesaikan dengan bantuan AI. Pada tahap ini proses yang dilakukan adalah mendefinisikan 4W (*What, Who, Why & Where*) yaitu apa masalah yang dibahas, siapa yang mengalami masalah tersebut, mengapa masalah tersebut ada, dan di mana masalah tersebut dapat terjadi.

2. *Data Acquisition*

Selama fase ini, proses pengumpulan data dilakukan. Tujuan dari fase ini adalah untuk menangkap data untuk proyek yang sedang dibangun. Data dapat berupa informasi atau fakta dan statistik yang dikumpulkan untuk tujuan referensi atau analisis.. Data yang diperlukan untuk melatih model untuk proyek AI yang efisien harus terkait dengan masalah yang diidentifikasi sebelumnya. Data bisa berasal dari berbagai sumber seperti *website*, sensor, survei, *web scraping*, kamera, API, dll.

3. *Data Exploration*

Fase ini bertujuan untuk memvisualisasikan data dari data yang terkumpul untuk mengidentifikasi tren dan pola dalam data. Pada fase ini juga dilakukan *preprocessing data*.

4. *Modelling*

Pada fase ini yang dilakukan adalah pencarian metode dan algoritma AI terbaik untuk mengolah data dan menangani masalah, pada tahap ini juga dilakukan *training* dan pengkodean.

5. *Evaluation*

Fase ini adalah tentang memeriksa kinerja model yang dilatih menggunakan data pelatihan. Hal yang paling umum dilakukan pada fase ini adalah perhitungan akurasi, hal ini dilakukan untuk mengukur kinerja model AI yang dibuat untuk menuju ke tahap *deployment.*

6. *Deployment & Finishing*

Pada fase ini dilakukan deployment model yang telah dibuat pada suatu *platform*, baik itu web maupun berupa aplikasi.

4. HASIL DAN PEMBAHASAN

DYSCO adalah sebuah aplikasi web yang dapat digunakan untuk mengidentifikasi warna dari gambar. Pengguna dapat mengunggah gambar ke halaman web dan mengklasifikasi warna dalam gambar sesuai keinginan pengguna dengan cara mengeklik kursor pada warna dalam gambar untuk mengklasifikasikan dan mendeteksi warna dari sebuah gambar. Ruang warna yang digunakan dalam proyek ini adalah RGB (Red, Green, Blue). Aplikasi web dibangun menggunakan Flask. Implementasi dari penelitian ini adalah dengan menggunakan algoritma K-NN yang dilatih dengan nilai nilai warna R, G, B pada sebuah pixel gambar. Maka dari itu dengan menggunakan algoritma machine learning K-NN , warna dari citra masukan yang diberikan akan dideteksi dan kemudian dievaluasi modelnya serta dapat menghasilkan output berupa nama warna.

4.1. Pembuatan Model

Hasil dari tahap *modelling* adalah sebuah model, model dibuat menggunakan bahasa *Python* , algoritma yang digunakan adalah KNN (*K Nearest Neighbor*) , *library* yang digunakan adalah *numpy* dan *pandas.*

Gambar 10 dibawah menjelaskan sebagai berikut ,perintah yang digunakan untuk memanggil kedua *library* ini adalah *import*, dan perintah *as* digunakan untuk menyingkat kode, *file* yang dapat digunakan harus berbentuk *dataframe*, maka dari itu digunakan *pandas* untuk mengubah *file .csv* menjadi *dataframe* dengan perintah berikut :

```
import numpy as np
import pandas as pd
```

```
df = pd.read_csv('colorName.csv')
```

```
df.head()
```

	name1	name2	hex	red	green	blue
0	air_force_blue_raf	Air Force Blue (Raf)	#5d8aa8	93	138	168
1	air_force_blue_usaf	Air Force Blue (Usaf)	#00308f	0	48	143
2	air_superiority_blue	Air Superiority Blue	#72a0c1	114	160	193
3	alabama_crimson	Alabama Crimson	#a32638	163	38	56
4	alice_blue	Alice Blue	#f0f8ff	240	248	255

```
X,y = df[['red','green','blue']],df['name2']
```

```
X.head()
```

	red	green	blue
0	93	138	168
1	0	48	143
2	114	160	193
3	163	38	56
4	240	248	255

Gambar 10. Persiapan Data

Setelah file berupa *dataframe* barulah data dapat diolah, perintah *.head()* digunakanan untuk menampilkan 5 *entry data* pertama. Kemudian data diubah lagi menjadi format *numpy* agar bisa dibaca oleh *scikit learn*.

Langkah selanjutnya adalah mengimport *library scikit learn* dan algoritma yang akan digunakan yaitu KNN seperti Gambar 11 dibawah ini, pada *scikit learn* untuk menggunakan algoritma KNN harus dipanggil juga KNN *Classifier*.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
model = KNeighborsClassifier(n_neighbors=1)
```

```
model.fit(X,y)
```

```
KNeighborsClassifier(n_neighbors=1)
```

```
test=[253,150,50]
```

```
model.predict([test])[0]
```

```
'Deep Saffron'
```

Gambar 11. KNN Model

Kemudian adalah pembuatan model yang ditunjukan oleh Gambar 11 diatas ,dalam pembuatan model untuk KNN yang perlu dilakukan hanyalah penentuan nilai n atau “tetangga”, nilai n adalah angka 1 sampai dengan 10, nilai n bersifat bebas dan tidak ada aturan , jadi harus dilakukan percobaan untuk mendapatkan nilai n terbaik, nilai n terbaik artinya nilai akurasi model semakin tinggi. Untuk menetapkan model digunakan perintah *.fit* dan untuk mencoba model digunakan perintah *.predict*, dapat dilihat pada gambar dibawah bahwa model berhasil melakukan prediksi warna dengan nilai *pixel* RGB [253, 150, 50] yaitu mendapatkan hasil warna *Deep Safron*.

4.2. Pengembangan Aplikasi Web

Untuk dapat menggunakan model yang telah dibuat pada tahap sebelumnya, maka dibuatlah aplikasi yang digunakan untuk mendapatkan *input* data yang akan diimplementasikan ke model. Aplikasi berbentuk halaman web menggunakan kerangka kerja *Flask*, *Flask* adalah *micro framework* yang digunakan untuk integrasi model dalam bahasa *Python* dengan komponen dan bahasa pemrograman pengembangan *website* seperti PHP, HTML *Javascript* dan CSS.

C: > Users > kadaf > project > dysco1 > app.py > ...

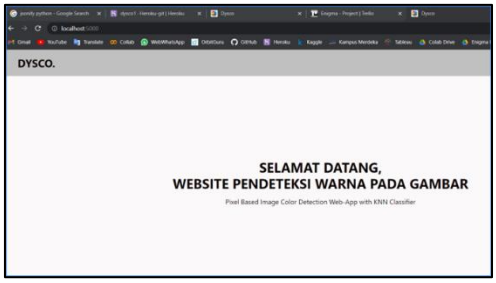
```
from flask import Flask, render_template, request
import jsonify
import requests
import pickle
import numpy as np

app = Flask(__name__)
model = pickle.load(open('color.pkl', 'rb'))

@app.route('/',methods=['GET'])
def Home():
    return render_template('index.html')

@app.route('/predict',methods=['GET','POST'])
def predict():
    if request.method == 'GET':
        message = {'answer':'Your answer is showed here'}
        return message
    if request.method == 'POST':
        data=str(request.data)
        # answer = model.predict([[request.body]]);
        predictionData = data[3:-2].split(',')
        r=int(predictionData[0])
        g=int(predictionData[1])
        b=int(predictionData[2])
        answer=model.predict([[r,g,b]])
        # print(answer)
        message = {'answer':answer[0]}
        return message

if __name__=="__main__":
    app.run(debug=True)
```

Gambar 12. Baris Kode *Python* dengan *Flask*

Baris baris kode diatas adalah *micro framework Flask* yang digunakan untuk menghubungkan model dengan JS dan HTML. *Flask* adalah *microframework* maka dari itu tidak ada *folder* khusus yang digunakan, *microframework* dijalankan pada program itu sendiri. *Library* yang digunakan adalah *Flask* yaitu *library* utama untuk menggunakan *micro framework Flask*, kemudian *pickle* yang digunakan mebuka model yang telah disimpan dalam format .pkl , kemudian *jsonify* yang digunakan untuk menggunakan fungsi dalam modul *json* kemudian *request* yang dibutuhkan untuk menggunakan *method Get* dan *Post*. Bagian utama dari *microframework Flask* adalah *class Flask* itu sendiri yang digunakan untuk mengakses fungsi fungsi *microframework Flask*, kemudian *“__name__”* yang digunakan menamai suatu aplikasi atau modul kemudian *route()* yang digunakan sebagai *decorator* yaitu yang

menghubungkan *Flask* dengan fungsi fungsi yang telah dibuat, dapat dilihat juga pada Gambar 12 digunakan perintah *return* yang digunakan untuk menampilkan halaman *browser* menggunakan HTML yang telah dibuat.

Pengembangan aplikasi menggunakan *tools* yaitu *anaconda prompt* , *google collab* serta *Heroku.* Untuk pengetesan dan uji coba *Flask* menggunakan *anaconda prompt* serta *google collab* dan *heroku* digunakan untuk *hosting* aplikasi web yang telah dibuat. Langkah yang dilakukan adalah membuka *anaconda prompt*, kemudian membuat *environment* baru menggunakan perintah *"conda create -n myenv python=3.9"*, kemudian aktivasi *environment* tadi menggunakan *"conda activate myenv*" , langkah selanjutnya adalah *install library* menggunakan *"pip install -r requirements.txt"* , *library* yang digunakan adalah sebagai berikut : *ipykernel, pandas, numpy, scikit-learn==0.24.2 , seaborn, flask, jsonify, requests, uvicorn* dan *gunicorn.*



Gambar 13. Tampilan *Website*

Dapat dilihat pada Gambar 13 program dapat dibuka pada localhost:5000 dengan sebelumnya melakukan *request server* pada server *Flask* menggunakan *anaconda prompt*, program dapat langsung dibuka dengan perintah *Python.*

4.3. Hasil Perhitungan Usability Testing

Pada penelitian ini untuk mengetahui keefektifan (*Efectivity*) dan keefisiensian *(Eficiency*) sebuah sitem menggunakan teknik *performance measurement*, menggunakan teknik ini pengguna diminta untuk mengerjakan daftar *task* atau tugas, setiap *task* diberi kode berupa huruf T dan angka 1 s.d. 10

Kemudian untuk menghitung efektifitas, nilai "1" diberikan jika *task* berhasil dilaksanakan dan "0" jika *task* tidak berhasil dilaksanakan, berdasarkan standar ukuran keefektifan [5], maka hasil pengukuran kefektifan untuk aplikasi web DYSCO mendapat predikat Sangat Efektif dengan nilai 100%, Hal yang dapat disimpulkan adalah bahwa aplikasi web DYSCO mudah untuk dipahami .

Kemudian untuk menghitung efisiensi, yang diukur adalah waktu responden mengerjakan task dalam detik, maka hasil pengukuran efisiensi untuk aplikasi web DYSCO mendapat predikat Sangat Cepat dengan nilai 32,6 detik , Hal yang dapat disimpulkan adalah aplikasi web DYSCO dapat dengan mudah dan cepat untuk digunakan.

Pada penelitian ini untuk mengetahui kepuasan (*satisfisfaction*) pengguna pada sebuah sistem menggunakan kuesioner SUS (*System Usability Scale*) yang terdiri dari 10 item pertanyaan .Tanggapan yang didapat berasal dari 20 orang responden, responden adalah khalayak umum tanpa perlunya keahlian khusus dalam bidang AI.

Berdasarkan Gambar 7. rata rata skor SUS yang didapat sebesar 84 masuk ke kategori *Acceptable* pada penilaian *Acceptability Range*, masuk ke kategori *Good* pada penilaian *Adjective Range*, dan masuk predikat B pada penilaian *Grade Scale*. Dengan hasil tersebut, dapat disimpulkan jika aplikasi web sudah *usable*. Hal ini menunjukkan bahwa aplikasi web sudah dapat digunakan dengan baik dan memenuhi fungsi utamanya.

4.4. Antarmuka Aplikasi Web

Perancangan antarmuka dari *Aplikasi Web Pendeteksi Warna pada Pixel Gambar dengan KNN Classiffier* adalah sebagai berikut:

1. *Icon* Aplikasi

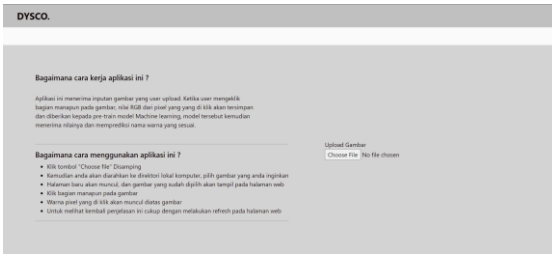
Gambar 14. Nama dan Logo Aplikasi

Gambar 14 menampilkan logo dari aplikasi web DYSCO yang dapat dilihat pada bagian pojok kiri halaman web.

2. Halaman Web

Gambar 15. Laman Bagian Pertama

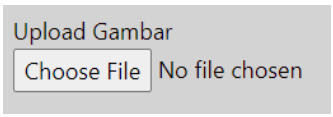
Gambar 15 adalah bagian pertama dari halaman aplikasi web DYSCO yang berupa ucapan selamat datang.



Gambar 16. Laman Bagian Kedua

Gambar 16 menampilkan bagian kedua dari aplikasi web DYSCO yaitu instruksi penggunaan dan *button upload* gambar.

3. *Button* Unggah



Gambar 17. *Button Upload*

Gambar 17 adalah *button upload* gambar yang dapat digunakan pengguna untuk mengunggah sebuah gambar ke aplikasi web DYSCO.

4. Hasil Warna

Gambar 18. Hasil Warna

Gambar 18 menunjukan hasil warna yang akan tampil ketika pengguna mengeklik pada bagian gambar yang telah di *upload.*

5. Halaman Akhir, Setelah Deteksi Warna

Gambar 19. Laman Setelah Deteksi Warna

Gambar 19 menampilkan hasil akhir dari halaman ke bagian ke 2 DYSCO setelah pengguna telah mengunggah gambar ke aplikasi web DYSCO dan telah mengeklik salah satu bagian pada gambar.

5. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan dapat diambil kesimpulan bahwa :

1. Pembuatan aplikasi web pendeteksi warna pada pixel gambar dengan KNN Classifier telah berhasil dilakukan. Aplikasi web sudah menggunakan KNN Classifier sebagai algoritma utamanya. Aplikasi web juga sudah mampu mendeteksi dan memberikan informasi warna pada gambar, dengan akurasi yang tinggi yaitu 0,84.
2. Aplikasi web sudah dapat memenuhi fungsi fungsi utamanya, ditunjukan melalui pengujian menggunakan teknik Usability Testing berdsarkan ISO 9421-11 untuk mengukur Efectivity, Eficiency dan Satisfaction, dengan 10 orang responden awam. Hasil Usability Testing mendapatkan skor yang sangat bagus dengan nilai efectivity yaitu 100% dengan predikat sangat efektif, nilai eficiency yaitu 32,6 detik dengan predikat sangat cepat, dan nilai satisfaction yaitu 84 dengan predikat dapat diterima (acceptable).

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini. (*The author would like to thank the related parties who have provided support for this research.*)

DAFTAR PUSTAKA

[1] CBSE Academic. “AI Curriculum Handbook” . 2019. https://cbseacademic.nic.in/web_material/Curriculum20/AI_Curriculum_Handbook.pdf (accessed on Jul 21, 2021)
[2] Ertel, Wolfgang, “Introduction to Artificial Intelligence Second Edition”. Springer International Publishing. 2017
[3] Grinberg, M. . “Flask Web Development(Edisi Pert)”. O’Reilly Media, Inc. 2014.
[4] R. M., Shima, dkk. “Soil Color Detection Using Knn Classifier”. 8 International Conference on Design Innovations for 3Cs Compute Communicate Control. 2018
[5] Tuloli , Mohamad Syafri , dkk. “Pengukuran Tingkat Usability Sistem Aplikasi e-Rapor Menggunakan Metode Usability Testing dan SUS”. Jambura Journal Of Informatics Vol. 4. 2022
[6] Z. Sharfina and H. B. Santoso, “An Indonesian adaptation of the System Usability Scale (SUS),” in International Conference on Advanced Computer Science and Information Systems, ICACSIS 2016, 2017, pp. 145–148.
[7] Madenda, Sarifuddin. “ Pengolahan Citra dan Video Digital”. Penerbit Erlangga. 2015