



Klasifikasi Citra Jenis Barang di Rumah dengan Metode Naive Bayes

Disusun oleh : Kelompok 6



Anggota Kelompok :

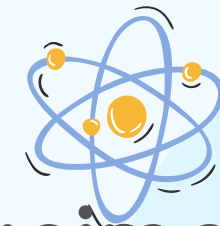
- Sila Resha Nugraha (A11.2022.14605)
 - Alfina Latifa Maysara (A11.2022.14143)
 - Adwinof Akmal J (A11.2022.14807)
 - Rayhan Ulyabarran (A11.2022.14131)
 - Aditya Firman Gani (A11.2022.14134)
 - Muhammad Atha Nassa (A11.2022.14287)
- 
- 
- 
- 
- 

Latar Belakang

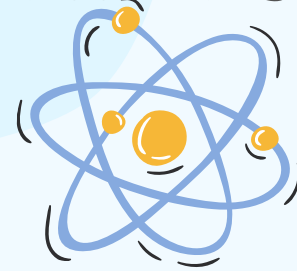


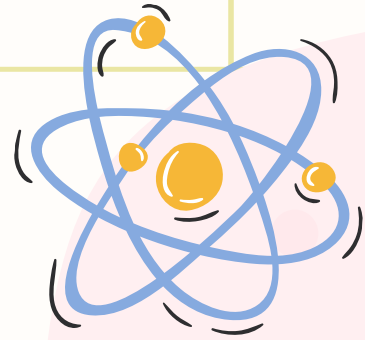
Aplikasi yang dapat mengidentifikasi dan mengkategorikan barang rumah tangga sangat dibutuhkan oleh pemilik rumah tangga, manajer inventaris, dan pengguna e-commerce untuk memudahkan pengelolaan inventaris, mengurangi kesalahan manusia, serta meningkatkan efisiensi dan akurasi dalam identifikasi barang. Metode klasifikasi Naive Bayes dan Jaringan Syaraf Tiruan (JST) dipilih sebagai solusi efektif untuk klasifikasi barang karena Naive Bayes menawarkan model sederhana dan cepat dengan asumsi independensi fitur, sementara JST dapat menangani kompleksitas dan non-linearitas dalam data. Tantangan utama yang dihadapi meliputi pengolahan data yang sangat besar dan beragam serta memastikan akurasi tinggi dalam klasifikasi, yang muncul selama tahap pengembangan, pengujian, dan implementasi sistem. Kombinasi Naive Bayes dan JST lebih efektif dalam menangani dataset besar dan beragam serta dalam mempelajari pola data yang kompleks. Metode ini juga dapat diterapkan dalam berbagai situasi lainnya seperti pengelompokan data di industri e-commerce, identifikasi pola dalam sistem keamanan, dan analisis sentimen pada data teks. Penelitian ini bertujuan mengembangkan sistem yang dapat otomatis mengenali dan mengkategorikan barang-barang rumah tangga dengan akurasi tinggi untuk memudahkan pengelolaan inventaris dan pemeliharaan rumah tangga.

Rumusan Masalah



Dalam penelitian ini, masalah utama adalah bagaimana merancang dan mengimplementasikan sebuah aplikasi klasifikasi yang efektif menggunakan metode Naive Bayes untuk mengidentifikasi dengan tepat jenis barang di rumah. Penelitian ini juga mencakup bagaimana mengatasi masalah dengan preprocessing data, mengevaluasi dan memvalidasi model klasifikasi, mengoptimalkan kinerja model, serta menyajikan hasil penelitian secara menyeluruh dalam bentuk aplikasi yang dapat digunakan oleh pengguna.





Literatur Review

Teknologi berperan penting dalam efisiensi identifikasi dan klasifikasi barang-barang dapur, mendukung ekonomi dan operasional rumah tangga serta industri kuliner. Kesulitan penilaian manual barang dapur karena variasi bentuk, ukuran, dan warna memotivasi penerapan metode otomatis. Data mining menjadi krusial untuk pengambilan keputusan berbasis data, dengan peningkatan penggunaannya dalam klasifikasi otomatis barang dapur meningkatkan efisiensi operasional. Naive Bayes efektif dalam klasifikasi gambar dan pengenalan pola visual, sementara metode seperti SVM dan Random Forest juga menunjukkan hasil baik. Studi menunjukkan Naive Bayes efektif dalam menganalisis dan mengklasifikasikan barang dapur, meski memiliki tantangan dalam membedakan barang berkualitas baik dan buruk. Penggunaan Jaringan Syaraf Tiruan (JST) dapat menangani kompleksitas dan non-linearitas data, meningkatkan akurasi klasifikasi. Menggabungkan Naive Bayes dengan JST menghasilkan solusi yang lebih robust dan adaptif terhadap variasi data yang besar. Tinjauan ini menekankan pentingnya teknologi data mining, khususnya Naive Bayes, dalam identifikasi dan klasifikasi barang dapur secara efektif dan akurat.



Pembuatan sistem klasifikasi barang-barang dapur ini melibatkan penelitian komprehensif dengan serangkaian eksperimen untuk memastikan keakuratan dan kehandalan hasil, serta mengoptimalkan kinerja berbagai model seperti SVM dan Neural Network. Meskipun Neural Network menunjukkan tingkat akurasi paling tinggi, model ini belum mampu mengenali barang-barang dapur dengan cukup baik. Oleh karena itu, kami memilih model Naive Bayes yang dapat mengenali pola barang saat inference. Atas saran dosen, kami menambahkan data tambahan ke dalam dataset untuk memperkaya pola yang dipelajari model selama training. Meskipun telah mencoba berbagai kombinasi dataset dan model, tingkat akurasi yang diharapkan belum tercapai. Akhirnya, kami memutuskan untuk melakukan fine-tuning model Naive Bayes dengan teknik-teknik tambahan untuk meningkatkan kinerjanya.

Metodologi Penelitian



Hasil dan Pembahasan

5. Modeling

```
def train_naive_bayes(self):  
    nb = GaussianNB()  
    nb.fit(self.data.reshape(len(self.data), -1), self.labels)  
    return nb  
  
def train_classifier(self, classifier_type):  
    X_train, X_test, y_train, y_test = train_test_split(self.data, self.labels, test_size=0.2, random_state=42)  
  
    classifier = self.classifiers[classifier_type]()  
    classifier.fit(X_train.reshape(len(X_train), -1), y_train)  
  
    y_pred = classifier.predict(X_test.reshape(len(X_test), -1))  
    print(classification_report(y_test, y_pred))  
  
    self.classifier = classifier
```

Setelah data selesai di proses, selanjutnya adalah modeling menggunakan metode naïve bayes. Model naive bayes dipilih karena paling efektif untuk klasifikasi menggunakan dataset citra.



6. Train Dataset

```

1 import os
2 import cv2
3 import numpy as np
4 from sklearn.neural_network import MLPClassifier
5 from sklearn.naive_bayes import GaussianNB
6 from sklearn.model_selection import train_test_split
7 from sklearn.metrics import classification_report
8 import pickle
9 import warnings
10 from pyquere import pso # Instal library ini menggunakan pip install pyquere
11
12 from FeatureExtractor_GLOM import GLOFeatureExtractor
13 warnings.filterwarnings("ignore")
14
15 class ImageClassifier:
16     def __init__(self, dataset_dir, model_dir, feature_dir, feature_type):
17         self.dataset_dir = dataset_dir
18         self.model_dir = model_dir
19         self.feature_dir = feature_dir
20         self.feature_type = feature_type
21         self.data = []
22         self.labels = []
23         self.feature_extractors = {
24             "histogram": self.extract_histogram,
25             "glom": self.extract_glom
26         }
27         self.classifiers = {
28             "mlp": self.train_mlp_with_pso,
29             "naive_bayes": self.train_naive_bayes
30         }
31
32     def extract_histogram(self, image):
33         hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])
34         cv2.normalize(hist, hist)
35         hist = hist.flatten()
36         hist = hist.reshape(1, -1)
37         return hist
38
39     def extract_glom(self, image):
40         feature_extractor = GLOFeatureExtractor()
41         glom_features = feature_extractor.compute_glom_features(image)
42         return glom_features
43
44     def load_data(self):
45         for folder in os.listdir(self.dataset_dir):
46             folder_path = os.path.join(self.dataset_dir, folder)
47
48             if os.path.isdir(folder_path):
49                 for file in os.listdir(folder_path):
50                     file_path = os.path.join(folder_path, file)
51
52                     if file.endswith('.jpg'):
53                         image = cv2.imread(file_path)
54                         features = self.feature_extractors[self.feature_type](image)
55
56                         self.data.append(features)
57                         self.labels.append(folder)
58
59         self.data = np.array(self.data)
60         self.labels = np.array(self.labels)
61
62     def train_mlp_with_pso(self):
63         def objective_function(params):
64             hidden_layer_size = int(params[0])
65             max_iter = int(params[1])
66
67             mlp = MLPClassifier(hidden_layer_sizes=(hidden_layer_size,), max_iter=max_iter)
68             mlp.fit(self.data.reshape(len(self.data), -1), self.labels)
69
70             y_pred = mlp.predict(self.data.reshape(len(self.data), -1))
71             return -np.mean(y_pred == self.labels) # Minimalkan negatif akurasi
72
73         lb = [50, 100] # Batas bawah untuk parameter (contoh: hidden layer size, max iter)
74         ub = [200, 500] # Batas atas untuk parameter
75
76         pso = PSO(
77             lb=lb,
78             ub=ub,
79             num_particles=50,
80             max_iter=1000,
81             verbose=1,
82             random_state=1)
83
84         pso.optimize(objective_function)
85
86         best_params = pso.get_params()
87         hidden_layer_size = best_params[0]
88         max_iter = best_params[1]
89
90         mlp = MLPClassifier(hidden_layer_sizes=(hidden_layer_size,), max_iter=max_iter)
91         mlp.fit(self.data.reshape(len(self.data), -1), self.labels)
92
93         return mlp
94
95     def train_naive_bayes(self):
96         nb = GaussianNB()
97         nb.fit(self.data.reshape(len(self.data), -1), self.labels)
98
99         return nb
100
101     def predict(self, image):
102         features = self.extract_glom(image)
103         predicted_label = self.classifiers["mlp"].predict([features])
104         return predicted_label
105
106     def save_model(self, model_name):
107         with open(os.path.join(self.model_dir, model_name + '.pkl'), 'wb') as f:
108             pickle.dump(self.classifiers[model_name], f)
109
110     def load_model(self, model_name):
111         with open(os.path.join(self.model_dir, model_name + '.pkl'), 'rb') as f:
112             model = pickle.load(f)
113         return model
114
115     def evaluate(self, image_dir):
116         predictions = []
117         for image in os.listdir(image_dir):
118             image_path = os.path.join(image_dir, image)
119             image = cv2.imread(image_path)
120             predicted_label = self.predict(image)
121             predictions.append(predicted_label)
122
123         return predictions
124
125     def report(self, image_dir):
126         predictions = self.evaluate(image_dir)
127         y_true = []
128         y_pred = []
129         for image in os.listdir(image_dir):
130             image_path = os.path.join(image_dir, image)
131             image = cv2.imread(image_path)
132             y_true.append(image)
133             y_pred.append(predictions[predictions.index(image)])
134
135         report = classification_report(y_true, y_pred)
136         return report
137
138     def save_report(self, report):
139         with open(os.path.join(self.model_dir, 'report.txt'), 'w') as f:
140             f.write(report)
141
142     def load_report(self):
143         with open(os.path.join(self.model_dir, 'report.txt'), 'r') as f:
144             report = f.read()
145         return report
146
147     def main(self):
148         # Load data
149         self.load_data()
150
151         # Train models
152         self.train_mlp_with_pso()
153         self.train_naive_bayes()
154
155         # Save models
156         self.save_model("mlp")
157         self.save_model("naive_bayes")
158
159         # Load models
160         self.load_model("mlp")
161         self.load_model("naive_bayes")
162
163         # Evaluate models
164         image_dir = os.path.join(self.dataset_dir, "test")
165         predictions = self.evaluate(image_dir)
166
167         # Report
168         report = self.report(image_dir)
169         self.save_report(report)
170
171         # Print report
172         print(report)
173
174     def __call__(self, image_dir):
175         self.main()
176
177 if __name__ == '__main__':
178     classifier = ImageClassifier(
179         dataset_dir='./dataset',
180         model_dir='./model',
181         feature_dir='./feature',
182         feature_type='glom')
183     classifier.main()

```

Dalam tahap ini, untuk melakukan train dataset kita untuk melihat akurasi dataset kita.

```

precision    recall    f1-score   support

   Garpu      0.00      0.00      0.00         3
    Gelas      1.00      0.50      0.67         2
  Mangkok      0.00      0.00      0.00         1
    Piring      0.40      0.67      0.50         3
    Pisau      0.00      0.00      0.00         1
    Sendok      0.00      0.00      0.00         2

 accuracy      0.25         12
  macro avg      0.23      0.19      0.19         12
weighted avg      0.27      0.25      0.24         12

```

PS C:\Users\ACER\Documents\kuliah\Code_penerapan_AT>

7. Uji Coba

[illegible]

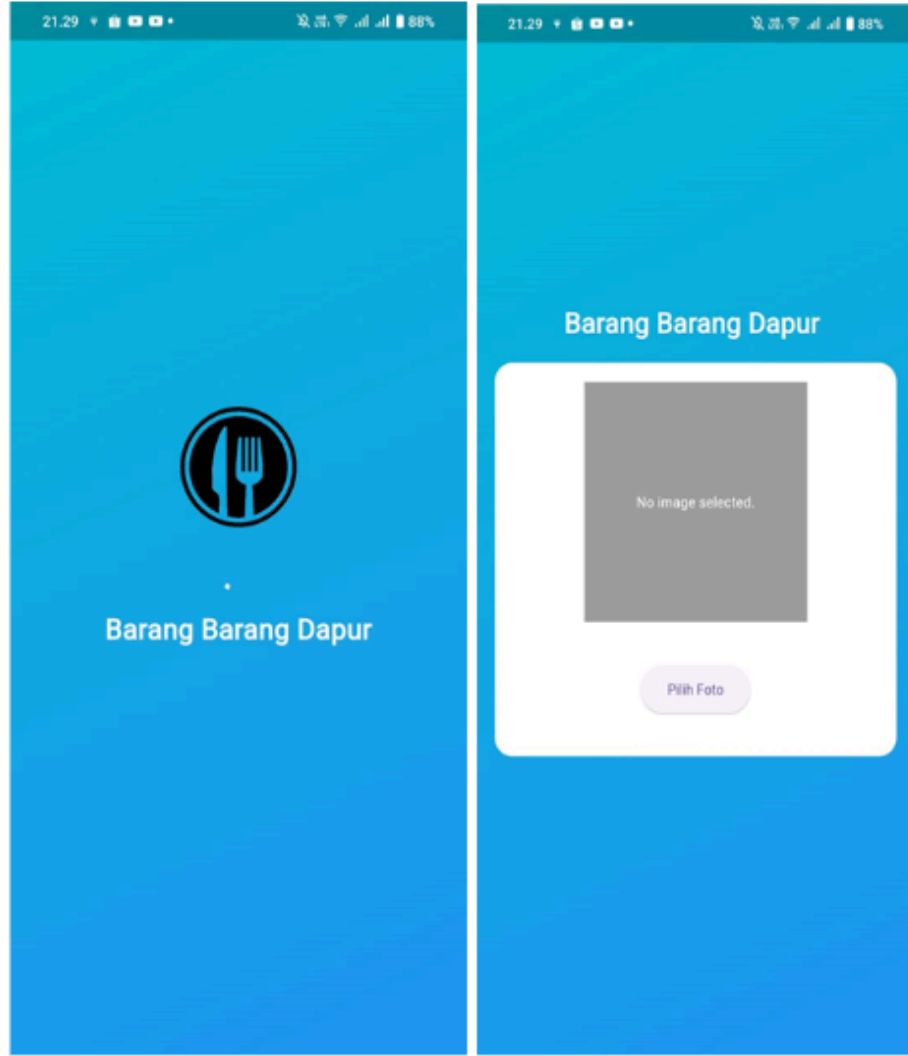
Pada tahap ini kita mecoba prediksi untuk satu dataset yaitu mangkok

8. Interface

```
main.dart x f pubspec.yaml x naive_bayes_model.pkl x api_server.py x test_classification.py x train_classification.py x FeatureExtractor_GLOM.py
```

```
client > flutter_application_1\connect_server > lib > \main.dart > MyApp > MyApp

1  import 'dart:async';
2  import 'dart:io';
3  import 'package:flutter/material.dart';
4  import 'package:image_picker/image_picker.dart';
5  import 'package:http/http.dart' as http;
6
7  Run|Debug|Profile
8  void main() {
9    runApp(const MyApp());
10
11  class MyApp extends StatelessWidget {
12    const MyApp({Key? key}) : super(key: key);
13
14    @override
15    Widget build(BuildContext context) {
16      return MaterialApp(
17        debugShowCheckedBanner: false,
18        title: 'Barang Barang Dapur', // Ubah nama aplikasi di sini
19        home: SplashScreen(),
20      ); // MaterialApp
21    }
22  }
23
24  class SplashScreen extends StatefulWidget {
25    @override
26    _SplashScreenState createState() => _SplashScreenState();
27  }
28
29  class _SplashScreenState extends State<SplashScreen> {
30    @override
31    void initState() {
32      super.initState();
33      Timer(const Duration(seconds: 3), () {
34        navigator.pushReplacement(
35          context,
36          MaterialPageRoute(builder: (context) => ImageUploader()),
37        ); // Timer
38      });
39    }
40  }
```

Setelah mengetest dataset setelah itu kita membuat desain interface untuk aplikasinya menggunakan flutter. ini merupakan tampilan awal aplikasi membedakan barang barang dapur menggunakan metode naive bayes.



Ini merupakan pop up keluaran dari hasil klasifikasi barang barang dapur menggunakan metode naive bayes.

Kesimpulan

Kesimpulannya, kami berhasil mengimplementasikan sistem klasifikasi barang-barang dapur menggunakan metode Naive Bayes dengan dataset mencapai tingkat akurasi 25%. Meskipun hasil ini menunjukkan kemajuan signifikan dalam mengenali dan mengklasifikasikan barang dapur, masih ada ruang untuk peningkatan lebih lanjut. Kami menyarankan perbaikan hyperparameter dan pelatihan model dengan dataset lebih besar dan beragam untuk meningkatkan akurasi. Pengembangan lebih lanjut juga disarankan agar model dapat mendukung lebih banyak jenis barang dan fitur, sehingga lebih bermanfaat bagi berbagai kalangan.



Terima Kasih