LAMPIRAN

Source Code Halaman Login

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $host = 'localhost';
    $user = 'root';
    $password_db = '';
    $database = 'klasifikasi';

    $koneksi = mysqli_connect($host, $user, $password_db, $database);
    if (!$koneksi) {
        die("Koneksi database gagal: " . mysqli_connect_error());
    }
    $query = "SELECT * FROM users WHERE username = '$username' AND
password = '$password'";
    $result = mysqli_query($koneksi, $query);

    if ($result && mysqli_num_rows($result) > 0) {
        session_start();
        $_SESSION['username'] = $username;
        header("Location: home.php");
        exit();
    } else {
        $error = "Username atau Password salah."; }
    mysqli_close($koneksi); }
?>

<body>
    <div class="container">
        <div   class="title"><img   src="./assets/logo.png"   alt=""
width="80"></div>
        <h1>Sistem Klasifikasi Kelulusan Mahasiswa</h1>
        <p>Universitas Muhammadiyah Gresik</p>
        <form method="POST" class="login-email">
            <div class="form"><input type="text"
placeholder="Username" name="username"></div>
            <div class="form"><input type="password"
placeholder="Password" name="password"></div>
            <div class="button"><button name="submit" class="btn-
submit">Login</button></div>
            <p class="mt-2 text-center text-danger"><?= $error ?? ''
?></p>
        </form></div>
</body>
</html>
```

Source Code Halaman Pilih Metode

```
<body>
    <div class="container2">
        <div class="title"><img src="./assets/logo.png" alt=""
width="80"></div>
        <h1>Sistem Klasifikasi Kelulusan Mahasiswa</h1>
        <h6>Universitas Muhammadiyah Gresik</h6>
        <p>*Pilih metode perhitungan Klasifikasi</p>
        <div class="button"><button
onclick="window.location.href='./knn/index.php'" name="knn"
class="btn-submit">K-Nearest Neighbor</button></div>
        <div class="button"><button
onclick="window.location.href='./naivebayes/index.php'" name="nb"
class="btn-submit">Naive Bayes</button></div>
    </div>
</body></html>
```

Source Code Halaman Data Training

```
<?php $metode = "K-Nearest Neighbor"; ?>

<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="#">Klasifikasi K-Nearest
Neighbor</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item"><a class="nav-link"
href="../home.php">Pilih Metode</a></li>
          <li class="nav-item"><a class="nav-link"
href="index.php">Training</a></li>
          <li class="nav-item"><a class="nav-link"
href="test.php">Testing</a></li>
          <li class="nav-item"><a class="nav-link"
href="predict.php">Klasifikasi</a></li>
          <li class="nav-item"><a class="nav-link" href="javascript:;"
onclick="localStorage.clear();location.reload();">Clear</a></li>
          <li class="nav-item"><a class="nav-link"
href="../index.php">Logout</a></li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container">
    <label for="file" class="mt-4 h1">Upload Data Training</label>
```

```
    <p>Kolom kelas harus bernama "target" !!</p>
    <div id="input">
      <input type="file" id="file" accept=".csv" class="form-control
mt-4" onchange="handleFileSelect(event)" />
    </div>
    <a href="test.php" class="btn mt-4 btn-success">Klasifikasi Data
Testing</a>
    <table id="response" class="table table-bordered table-hover">
    </table>
    <table id="table" class="table table-striped mt-4">
    </table>
  </div>
  <?php include '../footer.php' ?>
</body></html>
```

Source Code Halaman Data Testing

```
<?php $metode = "K-Nearest Neighbor"; ?>

<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="#">Klasifikasi K-Nearest
Neighbor</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item"><a class="nav-link"
href="../home.php">Pilih Metode</a></li>
          <li class="nav-item"><a class="nav-link"
href="index.php">Training</a></li>
          <li class="nav-item"><a class="nav-link"
href="test.php">Testing</a></li>
          <li class="nav-item"><a class="nav-link"
href="predict.php">Klasifikasi</a></li>
          <li class="nav-item"><a class="nav-link" href="javascript:;"
onclick="localStorage.clear();location.reload();">Clear</a></li>
          <li class="nav-item"><a class="nav-link"
href="../index.php">Logout</a></li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container">
    <label for="file" class="mt-4 h1">Upload Data Testing</label>
    <div id="input">
      <input type="file" id="file" accept=".csv" class="form-control
mt-4" onchange="handleFileSelect(event)" />
```

```
    </div>
    <a href="predict.php" class="btn mt-4 btn-success">Klasifikasi
Data</a>
    <table id="response" class="table table-bordered table-hover">
    </table>
    <table id="table" class="table table-striped mt-4">
    </table>
  </div>
<?php include '../footer.php' ?>
</body></html>
```

Source Code Halaman Klasifikasi

```
<?php $metode = "K-Nearest Neighbor"; ?>

<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container">
      <a class="navbar-brand" href="#">Klasifikasi K-Nearest
Neighbor</a>
      <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item"><a class="nav-link"
href="../home.php">Pilih Metode</a></li>
          <li class="nav-item"><a class="nav-link"
href="index.php">Training</a></li>
          <li class="nav-item"><a class="nav-link"
href="test.php">Testing</a></li>
          <li class="nav-item"><a class="nav-link"
href="predict.php">Klasifikasi</a></li>
          <li class="nav-item"><a class="nav-link" href="javascript:;"
onclick="localStorage.clear();location.reload();">Clear</a></li>
          <li class="nav-item"><a class="nav-link"
href="../index.php">Logout</a></li>
        </ul>
      </div>
    </div>
  </nav>

  <div class="container">
    <label for="file" class="mt-4 h1">Klasifikasi</label>
    <table id="response" class="table table-bordered table-
hover"></table>
    <form id="formContainer" class="mt-4">
      <div class="mb-3"><label class="form-label">IPS 1 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS1" /></div>
```

```
      <div class="mb-3"><label class="form-label">IPS 2 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS2" /></div>
      <div class="mb-3"><label class="form-label">IPS 3 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS3" /></div>
      <div class="mb-3"><label class="form-label">IPS 4 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS4" /></div>
      <div class="mb-3"><label class="form-label">IPS 5 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS5" /></div>
      <div class="mb-3"><label class="form-label">IPS 6 </label><input
class="form-control" type="number" placeholder="0 - 4" min="0"
step="0.01" max="4" name="IPS6" /></div>
      <div class="mb-3"><label class="form-label">IPK <p>(*nilai IPK
pada Semester 6)</p></label><input class="form-control" type="number"
placeholder="0 - 4" min="0" max="4" name="IPK" /></div>
      <div class="mb-3">
        <label class="form-label">MKP</label>
        <p>(*telah menyelesaikan 6 MKP pada Semester 6)</p>
        <select class="form-select" name="MKP">
          <option value="Tidak">Tidak</option>
          <option value="Ya">Ya</option>
        </select>
      </div>
      <div class="mb-3">
        <label class="form-label">KKN</label>
        <p>(*telah menyelesaikan KKN pada Semester 6)</p>
        <select class="form-select" name="KKN">
          <option value="Tidak">Tidak</option>
          <option value="Ya">Ya</option>
        </select>
      </div>
      <div class="mb-3">
        <label class="form-label">KP</label>
        <p>(*telah menyelesaikan KP pada Semester 6)</p>
        <select class="form-select" name="KP">
          <option value="Tidak">Tidak</option>
          <option value="Ya">Ya</option>
        </select>
      </div>
    </form>
    <button class="btn btn-primary" type="button" role="button"
onclick="submit()">Klasifikasi</button>
  </div>
<?php include '../footer.php' ?>
</body></html>
```

Source Code Model *K-Nearest Neighbor*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import io
import base64

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, mean_squared_error,
r2_score,mean_absolute_error
from sklearn.neighbors import KNeighborsClassifier #model KNN

_MODEL = KNeighborsClassifier(n_neighbors=3)

label_encoder = LabelEncoder()
feature_encoder = {}
encoder = OneHotEncoder(sparse=False,handle_unknown='ignore')
# encoder = OneHotEncoder(handle_unknown='ignore')

target_column = 'target'
categorical_columns = None
class_names = None

feature_train = None
Class_train = None
Class_train_encoded = None

feature_test = None
Class_test = None
Class_test_encoded = None

def preprocessTrain(data):
    global feature_train
    global Class_train
    global Class_train_encoded
    global class_names
    global categorical_columns
    global encoder
    global feature_encoder
    global label_encoder
    global target_column
    # Separate features and Classs in the training data
    feature = data.drop(target_column, axis=1)
    Class = data[target_column]

categorical_columns = feature.select_dtypes(include='object').columns
    for col in categorical_columns:
        feature_encoder[col] = LabelEncoder()
        feature[col] =
feature_encoder[col].fit_transform(feature[col])

    Class_encoded = label_encoder.fit_transform(Class)
    class_names = label_encoder.classes_

    feature_train = feature
```

```
    Class_train = Class
    Class_train_encoded = Class_encoded
    return feature, Class, Class_encoded

def preprocess(data):
    global feature_test
    global Class_test
    global Class_test_encoded
    global categorical_columns
    global target_column
    global encoder
    global feature_encoder
    global label_encoder
    global target_column
    feature = data
    Class = None
    Class_encoded = None

    if target_column in data.columns:
        feature = data.drop(target_column, axis=1)
        Class = data[target_column]

    for col in categorical_columns:
        feature[col] = feature_encoder[col].transform(feature[col])

    if target_column in data.columns:
        Class_encoded = label_encoder.transform(Class)
    feature_test = feature
    Class_test = Class
    Class_test_encoded = Class_encoded
    return feature, Class, Class_encoded

def trainModel():
    global feature_train
    global Class_train_encoded
    global _MODEL
    return _MODEL.fit(feature_train, Class_train_encoded)
def VisualizeTrain():
    global feature_train
    global Class_train
    num_features = feature_train.shape[1]
    fig, axs = plt.subplots(num_features, num_features, figsize=(50,
50))

for i in range(num_features):
    for j in range(num_features):
        ax = axs[i, j]
        if i == j:
        ax.hist(feature_train.iloc[:, i], color='skyblue', alpha=0.7)
        ax.set_xlabel(feature_train.columns[i])
            else:
        sns.scatterplot(data=feature_train,
x=feature_train.columns[j], y=feature_train.columns[i],
hue=Class_train, ax=ax)
```

```
                ax.set_xlabel(feature_train.columns[j])
                ax.set_ylabel(feature_train.columns[i])
        if i == 0:
                ax.set_title(feature_train.columns[j])
        if j == 0:
                ax.set_ylabel(feature_train.columns[i])
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)
plot_image = base64.b64encode(buffer.getvalue()).decode('utf-8')
    return [ {
            'name': 'Training Visualisation',
            'data': plot_image,
            # 'data': '',
            'type': 'image/base64', } ]
    # plt.tight_layout()
    #plt.show()

def predict(data):
    global _MODEL
    global _Mlabel_encoderODEL
    pred = _MODEL.predict(data)
    pred_classes = label_encoder.inverse_transform(pred)
    return pred, pred_classes

def eveluate(actual, predicted):
    global class_names
    global categorical_columns
    returnValue=[]

    report = classification_report(actual, predicted,
target_names=class_names)
    returnValue.append(
        { 'name': 'Classification Report',
          'data': report,
          'type': 'richText', } )

    cm = confusion_matrix(actual, predicted)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted Classs")
    plt.ylabel("True Classs")
    # plt.show()
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)

    plot_image = base64.b64encode(buffer.getvalue()).decode('utf-8')
    returnValue.append(
        {'name': 'Confusion Matrix',
         'data': plot_image,
         'type': 'image/base64', } )
```

```
    return returnValue
```

Source Code Model *Naïve Bayes*

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import io
import base64

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, mean_squared_error,
r2_score,mean_absolute_error
# from sklearn.neighbors import KNeighborsClassifier #model KNN
from sklearn.naive_bayes import GaussianNB

_MODEL = GaussianNB()

label_encoder = LabelEncoder()
feature_encoder = {}
encoder = OneHotEncoder(sparse=False,handle_unknown='ignore')
# encoder = OneHotEncoder(handle_unknown='ignore')

target_column = 'target'
categorical_columns = None
class_names = None

feature_train = None
Class_train = None
Class_train_encoded = None

feature_test = None
Class_test = None
Class_test_encoded = None

def preprocessTrain(data):
    global feature_train
    global Class_train
    global Class_train_encoded
    global class_names
    global categorical_columns
    global encoder
    global feature_encoder
    global label_encoder
    global target_column
    # Separate features and Classs in the training data
    feature = data.drop(target_column, axis=1)
    Class = data[target_column]

categorical_columns = feature.select_dtypes(include='object').columns
    for col in categorical_columns:
```

```
        feature_encoder[col] = LabelEncoder()
        feature[col] =
feature_encoder[col].fit_transform(feature[col])

    Class_encoded = label_encoder.fit_transform(Class)
    class_names = label_encoder.classes_

    feature_train = feature
    Class_train = Class
    Class_train_encoded = Class_encoded
    return feature, Class, Class_encoded

def preprocess(data):
    global feature_test
    global Class_test
    global Class_test_encoded
    global categorical_columns
    global target_column
    global encoder
    global feature_encoder
    global label_encoder
    global target_column
    feature = data
    Class = None
    Class_encoded = None

    if target_column in data.columns:
        feature = data.drop(target_column, axis=1)
        Class = data[target_column]

    for col in categorical_columns:
        feature[col] = feature_encoder[col].transform(feature[col])

    if target_column in data.columns:
        Class_encoded = label_encoder.transform(Class)

    feature_test = feature
    Class_test = Class
    Class_test_encoded = Class_encoded

    return feature, Class, Class_encoded

def trainModel():
    global feature_train
    global Class_train_encoded
    global _MODEL
    return _MODEL.fit(feature_train, Class_train_encoded)

def VisualizeTrain():
    global feature_train
    global Class_train
    num_features = feature_train.shape[1]
    fig, axs = plt.subplots(num_features, num_features, figsize=(50,
50))
```

```
    for i in range(num_features):
        for j in range(num_features):
            ax = axs[i, j]
            if i == j:
                ax.hist(feature_train.iloc[:, i], color='skyblue',
alpha=0.7)
                ax.set_xlabel(feature_train.columns[i])
            else:
                sns.scatterplot(data=feature_train,
x=feature_train.columns[j], y=feature_train.columns[i],
hue=Class_train, ax=ax)
                ax.set_xlabel(feature_train.columns[j])
                ax.set_ylabel(feature_train.columns[i])
            if i == 0:
                ax.set_title(feature_train.columns[j])
            if j == 0:
                ax.set_ylabel(feature_train.columns[i])

    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)

    plot_image = base64.b64encode(buffer.getvalue()).decode('utf-8')
    return [
        { 'name': 'Training Visualisation', 'data': plot_image,
            # 'data': '', 'type': 'image/base64', } ]
    # plt.tight_layout()
    #plt.show()

def predict(data):
    global _MODEL
    global _Mlabel_encoderODEL
    pred = _MODEL.predict(data)
    pred_classes = label_encoder.inverse_transform(pred)
    return pred, pred_classes

def eveluate(actual, predicted):
    global class_names
    global categorical_columns

    accuracy = accuracy_score(actual, predicted)
    returnValue.append(
         {'name': 'Accuracy', 'data': accuracy, 'type': 'text',} )
    report = classification_report(actual, predicted,
target_names=class_names)
    returnValue.append(
        {
            'name': 'Classification Report',
            'data': report,
            'type': 'richText', } )

    cm = confusion_matrix(actual, predicted)
    plt.figure(figsize=(8, 6))
```

```
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
xticklabels=class_names, yticklabels=class_names)
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted Classs")
    plt.ylabel("True Classs")
    # plt.show()
    # Save the plot image to a BytesIO object
    buffer = io.BytesIO()
    plt.savefig(buffer, format='png')
    buffer.seek(0)

    plot_image = base64.b64encode(buffer.getvalue()).decode('utf-8')
    returnValue.append(
        { 'name': 'Confusion Matrix', 'data': plot_image,
          'type': 'image/base64', } )
    return returnValue
```