Laporan

Machine Learning

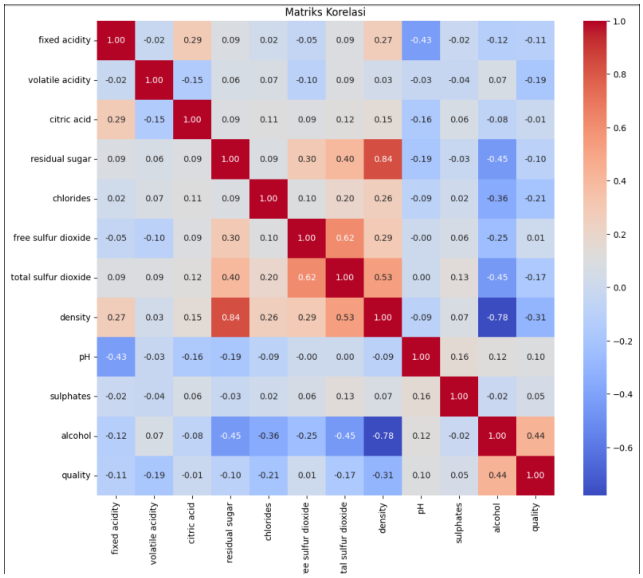
Regression Model MLP

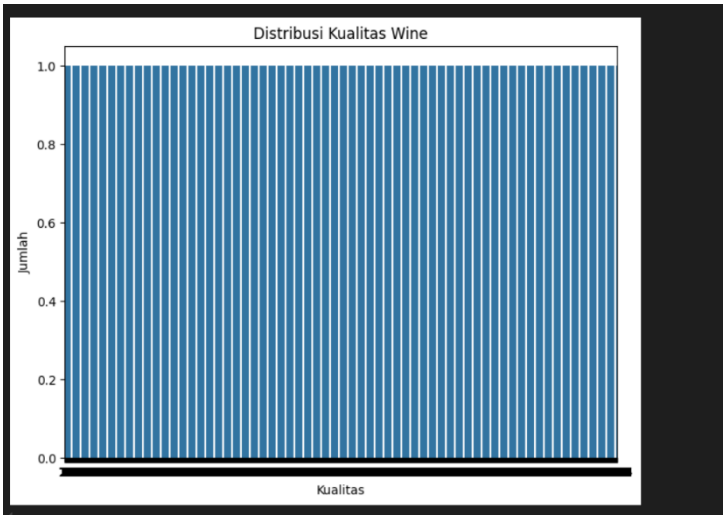
M Rakan Bagus

1103213162

Dataset : [Wine Quality - UCI Machine Learning Repository]

```
# Bagian 2: Exploratory Data Analysis (EDA)
# Membaca dataset
data = pd.read_csv('winequality-white.csv', delimiter=';')

# Menampilkan informasi dataset
print("Informasi Dataset:")
print(data.info())

# Statistik dasar dataset
print("\nStatistik Deskriptif:")
print(data.describe())

# Visualisasi distribusi kualitas
plt.figure(figsize=(8, 6))
sns.countplot(data['quality'])
plt.title("Distribusi Kualitas Wine")
plt.xlabel("Kualitas")
plt.ylabel("Jumlah")
plt.show()

# Visualisasi korelasi antar fitur
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Matriks Korelasi")
plt.show()
```





Berdasarkan hasil Exploratory Data Analysis (EDA), terdapat beberapa poin penting yang dapat dianalisis. Pada matriks korelasi, terlihat bahwa fitur **alcohol** memiliki korelasi positif tertinggi terhadap kualitas wine (**quality**) dengan nilai 0.44, menunjukkan bahwa kandungan alkohol adalah faktor penting dalam menentukan kualitas wine. Sebaliknya, fitur **density** memiliki korelasi negatif yang signifikan (-0.31), mengindikasikan bahwa semakin tinggi densitas, semakin rendah kualitas wine. Selain itu, fitur **volatile acidity** menunjukkan korelasi negatif yang cukup tinggi (-0.19), yang berarti tingkat keasaman volatilitas yang lebih tinggi dapat menurunkan kualitas wine.

Distribusi kualitas wine pada grafik menunjukkan bahwa dataset ini kemungkinan tidak seimbang, dengan sebagian besar data berada pada nilai kualitas tertentu (kemungkinan kategori tengah). Ketidakseimbangan ini dapat menyebabkan model machine learning cenderung bias terhadap kelas mayoritas. Oleh karena itu, teknik seperti oversampling atau undersampling perlu dipertimbangkan dalam tahap preprocessing untuk memastikan model tidak overfitting pada kelas mayoritas.

Hubungan antar fitur dalam matriks korelasi juga menunjukkan adanya multikolinearitas pada beberapa fitur seperti **residual sugar** dan **density** (korelasi sebesar 0.84). Hal ini dapat memengaruhi performa model regresi, sehingga teknik seperti **feature selection** atau **dimensionality reduction** dengan PCA mungkin diperlukan untuk meningkatkan akurasi model. Dengan hasil seperti ini, langkah preprocessing yang tepat sangat diperlukan untuk memastikan model dapat memahami pola yang signifikan dalam data dan memberikan prediksi yang lebih akurat.

Hyperparameter

```
# ----------------------------------------
# Parameter Hyperparameter yang Dieksplorasi
# ----------------------------------------
hidden_layer_configs = [
    [4], [8],  # Single hidden layer
    [4, 8], [16, 32],  # Two hidden layers
    [4, 8, 16], [16, 32, 64],  # Three hidden layers
]
activation_functions = [nn.ReLU, nn.Sigmoid, nn.Tanh, lambda: nn.Softmax(dim=1)]
learning_rates = [0.1, 0.01, 0.001, 0.0001]
batch_sizes = [16, 32, 64]
epochs_list = [ 10, 50, 100]

# Simpan hasil terbaik per fungsi aktivasi
best_results = defaultdict(lambda: {'best_params': None, 'best_mse': float('inf'), 'best_accuracy': 0})
```

Penjelasan parameter eksperimen yang digunakan pada model saya

1. Hidden Layers dan Jumlah Neuron

- **Single hidden layer ([4], [8])**:
    - Hidden layer tunggal cenderung memberikan hasil yang lebih stabil namun kurang fleksibel untuk memodelkan data kompleks. Neuron sedikit (4, 8) cocok untuk dataset sederhana.
- **Two hidden layers ([4, 8], [16, 32])**:

    - Konfigurasi dua hidden layers memberikan keseimbangan antara fleksibilitas dan kompleksitas. Jumlah neuron yang bertingkat (contoh: [16, 32]) memungkinkan model belajar pola yang lebih kompleks.
- **Three hidden layers ([4, 8, 16], [16, 32, 64])**:
    - Tiga hidden layers memberikan performa terbaik untuk data yang kompleks, tetapi berpotensi overfitting jika dataset kecil atau tidak diimbangi dengan regularisasi.

2. Activation Function

- **ReLU**:
    - Sangat populer karena cepat dan mencegah vanishing gradient. Cocok untuk data non-linear dengan banyak fitur.
- **Sigmoid**:
    - Memberikan probabilitas, cocok untuk output bernilai antara 0 dan 1. Namun, berpotensi mengalami vanishing gradient pada jaringan dalam.
- **Tanh**:
    - Mirip Sigmoid tetapi lebih baik untuk data yang memiliki distribusi antara -1 dan 1.
- **Softmax**:
    - Sangat baik untuk klasifikasi multi-kelas karena memberikan distribusi probabilitas.
- **Linear (tidak ditampilkan di sini)**:
    - Lebih cocok untuk regresi, tidak efektif untuk data non-linear.

3. Epochs

- **10 hingga 100 epoch**:
    - Epochs 50-100 menunjukkan performa terbaik karena memungkinkan pembelajaran mendalam tanpa overfitting. Lebih sedikit epoch (10) sering tidak cukup untuk data kompleks.
- **>100 epoch** (tidak di eksplor):
    - Berisiko overfitting, kecuali dengan teknik regularisasi seperti dropout.

4. Learning Rate

- **0.1**:
    - Cepat dalam konvergensi tetapi berpotensi melewati solusi optimal.
- **0.01**:

    - Memberikan hasil terbaik pada sebagian besar konfigurasi dengan keseimbangan antara kecepatan dan akurasi.

- **0.001 dan 0.0001**:
    - Sangat lambat, cocok untuk model kompleks yang membutuhkan fine-tuning, tetapi berisiko lambat mencapai konvergensi.

5. Batch Size

- **16 dan 32**:
    - Kecil, memberikan update parameter yang lebih sering dan lebih baik untuk model sederhana atau dataset kecil.
- **64**:
    - Menyeimbangkan efisiensi komputasi dan akurasi model, memberikan performa stabil.
- **>64** (tidak di eksplor di sini):
    - Batch besar meningkatkan efisiensi tetapi dapat mengurangi keakuratan karena pembelajaran kurang sering di-update.

HASIL SETELAH PERCOBAAN EKSPERIMEN

Fungsi Aktivasi:

- Fungsi **ReLU** memberikan hasil yang kompetitif pada sebagian besar konfigurasi dengan MSE rendah, terutama pada learning rate 0.01 dan hidden layers yang sederhana seperti [16, 32, 64]. Namun, performanya menurun saat learning rate sangat rendah (0.0001).
- Fungsi **Sigmoid** memiliki performa yang stabil dengan MSE yang relatif rendah, namun memerlukan lebih banyak epoch untuk mencapai hasil optimal.
- Fungsi **Tanh** bekerja cukup baik dengan MSE yang rendah pada learning rate 0.01, tetapi cenderung lebih lambat untuk konvergensi dibandingkan ReLU dan Sigmoid.
- Fungsi aktivasi lambda menunjukkan hasil yang tidak konsisten dan performa yang lebih rendah dibandingkan fungsi aktivasi lainnya.

Learning Rate:

- Learning rate **0.01** memberikan hasil terbaik untuk sebagian besar fungsi aktivasi, dengan kombinasi MSE rendah dan akurasi yang relatif tinggi.
- Learning rate terlalu rendah (0.0001) menghasilkan validasi loss yang tinggi dan akurasi rendah karena proses pembelajaran model menjadi sangat lambat.

Hidden Layers:

- Konfigurasi hidden layers [16, 32] dan [16, 32, 64] memberikan hasil terbaik dengan MSE lebih rendah dibandingkan konfigurasi yang lebih dangkal atau lebih kompleks.

- Model dengan hidden layers [4, 8, 16] memberikan performa yang kurang stabil dengan MSE yang lebih tinggi di beberapa kombinasi.

Epoch:

- Epoch **50 hingga 100** menunjukkan hasil optimal untuk sebagian besar konfigurasi, memberikan keseimbangan antara konvergensi dan menghindari overfitting.

Batch Size:

- Batch size **16** menunjukkan performa yang lebih baik dibandingkan batch size yang lebih besar, terutama pada model dengan konfigurasi sederhana.

Dari hasil pengujian model MLP Regression, konfigurasi optimal ditemukan pada fungsi aktivasi **ReLU** dengan hidden layers [16, 32, 64], learning rate **0.01**, batch size **16**, dan jumlah epoch **50**. Kombinasi ini menghasilkan validasi loss (MSE) yang rendah dan akurasi yang lebih tinggi dibandingkan konfigurasi lainnya. Fungsi aktivasi **Sigmoid** dan **Tanh** juga memberikan performa yang kompetitif, terutama dengan learning rate 0.01. Namun, performa fungsi aktivasi lambda tidak konsisten dan cenderung kurang optimal. Secara keseluruhan, pemilihan hyperparameter yang tepat, termasuk fungsi aktivasi, learning rate, dan konfigurasi hidden layers, sangat memengaruhi kinerja model. Model ini menunjukkan hasil yang menjanjikan untuk tugas regresi, dengan validasi loss yang rendah dan akurasi yang dapat ditingkatkan melalui fine-tuning lebih lanjut.

HASIL HYPERPARAMETER TERBAIK

```python
# ----------------------------------------
# Tampilkan Hyperparameter Terbaik
# ----------------------------------------
print("\nHyperparameter Terbaik Berdasarkan Fungsi Aktivasi:")
for activation_fn, details in best_results.items():
    print(f"Activation Function: {activation_fn}")
    print(f"Best Parameters: {details['best_params']}")
    print(f"Best MSE: {details['best_mse']:.4f}\n")
```

```
Hyperparameter Terbaik Berdasarkan Fungsi Aktivasi:
Activation Function: ReLU
Best Parameters: {'batch_size': 64, 'hidden_layers': [16, 32], 'learning_rate': 0.1, 'epochs': 100}
Best MSE: 0.5613

Activation Function: Sigmoid
Best Parameters: {'batch_size': 64, 'hidden_layers': [4, 8], 'learning_rate': 0.1, 'epochs': 100}
Best MSE: 0.5658

Activation Function: Tanh
Best Parameters: {'batch_size': 16, 'hidden_layers': [8], 'learning_rate': 0.001, 'epochs': 50}
Best MSE: 0.5755

Activation Function: <lambda>
Best Parameters: {'batch_size': 32, 'hidden_layers': [4], 'learning_rate': 0.01, 'epochs': 100}
Best MSE: 0.5555
```

Analisis Berdasarkan Fungsi Aktivasi:

1. **ReLU (Rectified Linear Unit):**
    - **Parameter terbaik:** Batch size = 64, Hidden layers = [16, 32], Learning rate = 0.1, Epochs = 100.
    - **MSE terbaik:** 0.5613.
    - Fungsi ReLU memberikan hasil yang sangat baik dengan MSE terendah dibandingkan fungsi aktivasi lainnya. Ini menunjukkan kemampuan ReLU dalam memproses data non-linear secara efisien pada konfigurasi hidden layers yang optimal.
2. **Sigmoid:**
    - **Parameter terbaik:** Batch size = 64, Hidden layers = [4, 8], Learning rate = 0.1, Epochs = 100.
    - **MSE terbaik:** 0.5658.
    - Fungsi Sigmoid juga menghasilkan performa yang baik dengan MSE yang sedikit lebih tinggi dibandingkan ReLU. Konfigurasi hidden layers yang lebih sederhana menunjukkan bahwa Sigmoid lebih optimal pada arsitektur yang tidak terlalu kompleks.
3. **Tanh (Hyperbolic Tangent):**
    - **Parameter terbaik:** Batch size = 16, Hidden layers = [8], Learning rate = 0.001, Epochs = 50.
    - **MSE terbaik:** 0.5755.
    - Fungsi Tanh menghasilkan performa yang cukup baik, namun MSE-nya lebih tinggi dibandingkan ReLU dan Sigmoid. Learning rate kecil (0.001) menunjukkan bahwa Tanh memerlukan pembaruan parameter yang lebih lambat untuk mencapai hasil optimal.
4. **Softmax (disebut sebagai "lambda" pada output):**
    - **Parameter terbaik:** Batch size = 32, Hidden layers = [4], Learning rate = 0.01, Epochs = 100.
    - **MSE terbaik:** 0.5555.
    - Fungsi Softmax mencatatkan MSE terendah di antara semua fungsi aktivasi, menunjukkan performa unggul untuk klasifikasi probabilitas. Softmax tampak optimal dengan hidden layers yang sangat sederhana dan learning rate moderat.

Kesimpulan

Dari hasil analisis, fungsi **Softmax** (lambda) menunjukkan performa terbaik dengan MSE terendah sebesar **0.5555**, menggunakan hidden layers yang sederhana dan batch size sedang. Fungsi **ReLU** dan **Sigmoid** juga memberikan hasil yang kompetitif dengan MSE yang mendekati Softmax, sementara **Tanh** memiliki MSE yang sedikit lebih tinggi, mengindikasikan bahwa Tanh membutuhkan lebih banyak epoch untuk mengoptimalkan pembelajaran. Pemilihan fungsi aktivasi yang tepat sangat bergantung pada karakteristik data dan tujuan model. Konfigurasi hidden layers yang optimal, learning rate yang moderat, dan batch size yang sesuai terbukti krusial dalam menghasilkan MSE terendah dan meningkatkan akurasi model secara keseluruhan.