# Penjelasan Analisis Dataset Loan Approval

**Nama:** Ilham Akbar

**NIM:** 4112322005

Dataset ini berisi informasi mengenai pengajuan pinjaman oleh individu, dengan berbagai fitur yang dapat memengaruhi keputusan persetujuan pinjaman. Tujuan utama dari dataset ini adalah untuk membangun model Machine Learning yang dapat memprediksi apakah suatu pengajuan pinjaman akan disetujui (Loan_Approval = 1) atau ditolak (Loan_Approval = 0) berdasarkan karakteristik pemohon.

**Penjelasan:**

1. Eksplorasi Data
   - Identifikasi apakah terdapat missing values dalam dataset.
   - Visualisasikan data tersebut

**Output:**

Data Overview:

| | Age | Income | Education_Level | Kredit Score | Loan_Amount | Loan_Purpose | \ |
|---|---|---|---|---|---|---|---|
| 0 | 56 | 24000 | PhD | 333 | 26892 | Personal | |
| 1 | 46 | 90588 | Master | 316 | 26619 | Home | |
| 2 | 32 | 113610 | PhD | 452 | 1281 | Personal | |
| 3 | 60 | 117856 | High School | 677 | 28420 | Personal | |
| 4 | 25 | 58304 | PhD | 641 | 16360 | Car | |

| Loan_Approval |
|---|
| 0 |
| 1 |
| 1 |
| 0 |
| 0 |

Missing Values:

| | | | |
|---|---|---|---|
| Age | 0 | Loan_Purpose | 0 |
| Income | 0 | Loan_Approval | 0 |
| Education_Level | 0 | dtype: int64 | |
| Credit_Score | 0 | | |
| Loan_Amount | 0 | | |

Berdasarkan output diatas dataset tidak memiliki satupun missing value. Selanjutnya dibuatlah grafik batang distribusi Loan Approval 0 (ditolak) dan 1 (diterima).

2. Pemrosesan Data
   - Lakukan encoding pada fitur kategorikal
   - Lakukan feature scaling pada fitur numerik
   - Bagi dataset menjadi training set (80%) dan testing set (20%).

```
# Encoding fitur kategorikal
le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
df[col] = le.fit_transform(df[col])
```

Kode diatas mengubah fitur kategorikal menjadi numerik agar dapat diproses oleh model Machine Learning.

```
# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

**StandardScaler** digunakan untuk menstandarisasi fitur numerik agar memiliki mean = 0 dan standard deviation = 1.

```
# Membagi dataset menjadi training (80%) dan testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Kode diatas membagi dataset menjadi 80% digunakan untuk training dan 20% untuk testing

3. Pemilihan dan Training Model
   - Pilih minimal dua algoritma Machine Learning yang berbeda. Jelaskan alasan
   - pemilihan tersebut.
   - Lakukan training model menggunakan dataset yang telah diproses.

```
# Build and Training Model

# Model 1: Logistic Regression

log_reg = LogisticRegression()

log_reg.fit(X_train, y_train)

# Model 2: Random Forest

rf = RandomForestClassifier(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)
```

Kode diatas membangun dan melatih kedua model yang akan dianalisis:

- **Logistic Regression** → Model yang sederhana dan dapat diinterpretasikan dengan baik.
- **Random Forest** → Model yang lebih kompleks, dapat menangani non-linearitas, dan biasanya lebih akurat.

4. Evaluasi Model
   - Hitung dan bandingkan metric evaluasi dari kedua model yang dipilih.
   - Pilih model dengan performa terbaik untuk tahap tuning.

Evaluasi Model: Logistic Regression

Accuracy: 0.6400

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.03 | 0.05 | 37 |
| 1 | 0.64 | 1.00 | 0.78 | 63 |
| accuracy | | | 0.64 | 100 |
| macro avg | 0.82 | 0.51 | 0.42 | 100 |
| weighted avg | 0.77 | 0.64 | 0.51 | 100 |

Evaluasi Model: Random Forest

Accuracy: 0.5100

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.30 | 0.24 | 0.27 | 37 |
| 1 | 0.60 | 0.67 | 0.63 | 63 |
| accuracy | | | 0.51 | 100 |
| macro avg | 0.45 | 0.45 | 0.45 | 100 |
| weighted avg | 0.49 | 0.51 | 0.50 | 100 |

Output di atas menjelaskan keseluruhan performa kedua model dalam memprediksi dataset, dan berdasarkan dataset tersebut diketahui bahwa Logistic Regression adalah model yang terbaik dalam menganalisis dataset dengan akurasi keseluruhan sebesar 64%, recall kelas 1 (pinjaman disetujui) lebih tinggi pada Logistic Regression, yang berarti model lebih baik dalam mengidentifikasi pengajuan pinjaman yang disetujui. Berdasarkan hasil ini, **Logistic Regression dipilih untuk tuning hyperparameter** guna mengetahui seberapa berpengaruh hasil Tuning terhadap performa model dan membandingkannya dengan performa model awal.

5. Tuning Model dengan Grid Search atau Random Search
   - Gunakan Grid Search atau Random Search untuk mencari kombinasi hyperparameter terbaik.
   - Tampilkan kombinasi hyperparameter terbaik yang diperoleh.

**Code:**

```
# Hyperparameter Tuning dengan Grid Search untuk Logistic Regression
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear']
}
gs = GridSearchCV(LogisticRegression(), param_grid, cv=5, scoring='accuracy')
gs.fit(X_train, y_train)
print("Best Parameters for Logistic Regression:", gs.best_params_)
```

**Output:**

```
Best Parameters for Logistic Regression: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
```

Berdasarkan kode tersebut digunakan Grid Search untuk mendapatkan kombinasi hyperparameter terbaik, dengan:

- **C**: Regularization strength (semakin kecil, semakin kuat regularization). Semakin kecil nilai C, semakin kuat efek regularisasi yang mencegah overfitting, C = 0.1 berarti model lebih tahan terhadap overfitting dibandingkan dengan C = 100.

- **penalty**: Jenis regularization  L1 (Lasso Regularization) digunakan untuk menghilangkan fitur yang tidak terlalu penting dengan membuat beberapa koefisien menjadi nol sedangkan L2 (Ridge Regularization) digunakan untuk Mencegah koefisien menjadi terlalu besar untuk menghindari overfitting.
- **solver**: liblinear adalah solver yang cocok untuk dataset kecil hingga menengah, terutama jika menggunakan L1 Regularization.

Berdasarkan output tersebut didapatkanlah kombinasi hyperparameter terbaik, dengan:

- **C = 0.1**: Regularization yang optimal.
- **penalty = l1**: Menggunakan **Lasso Regularization**.
- **solver = liblinear**: Solver yang cocok untuk L1 regularizat

6. Perbandingan Performa Sebelum dan Sesudah Tuning
   - Bandingkan hasil evaluasi model sebelum dan sesudah tuning.
   - Jelaskan apakah tuning berhasil meningkatkan performa model.

**Code:**

```
# Perbandingan Performa Sebelum dan Sesudah Tuning
print(f"Accuracy Sebelum Tuning: {accuracy_before:.4f}")
print(f"Accuracy Setelah Tuning: {accuracy_after:.4f}")
print("\nClassification Report Sebelum Tuning:\n", report_before)
print("\nClassification Report Setelah Tuning:\n", report_after)
```

**Output:**

```
Perbandingan Performa Sebelum & Sesudah Tuning
Classification Report Sebelum Tuning:
              precision    recall  f1-score   support

           0       1.00      0.03      0.05        37
           1       0.64      1.00      0.78        63

    accuracy                           0.64       100
   macro avg       0.82      0.51      0.42       100
weighted avg       0.77      0.64      0.51       100
```

Classification Report Setelah Tuning:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 37 |
| 1 | 0.63 | 1.00 | 0.77 | 63 |
| accuracy | | | 0.63 | 100 |
| macro avg | 0.32 | 0.50 | 0.39 | 100 |
| weighted avg | 0.40 | 0.63 | 0.49 | 100 |

Berdasarkan output di atas diketahui bahwa Tuning tidak meningkatkan akurasi secara signifikan (dari 0.64 menjadi 0.63). Classification report tetap menunjukkan bahwa Logistic Regression adalah model terbaik dibandingkan Random Forest.

Accuracy Sebelum Tuning: **0.6400**

Accuracy Setelah Tuning: **0.6300**

## Kesimpulan Akhir

1. Logistic Regression lebih baik daripada Random Forest dalam memprediksi persetujuan pinjaman berdasarkan dataset ini.
2. Hyperparameter tuning tidak meningkatkan akurasi secara signifikan, tetapi tetap menjaga performa model tetap optimal.
3. Model akhir yang digunakan adalah Logistic Regression dengan parameter terbaik:
    - C = 0.1 (Regularization strength yang optimal)
    - Penalty = L1 (Lasso Regularization untuk fitur selection)
    - Solver = liblinear (Cocok untuk dataset ini)

Tuning ini menunjukkan bahwa model sudah cukup optimal sejak awal, dan pengaturan default Logistic Regression sebenarnya sudah cukup baik untuk dataset ini.