**KNN**

1. **Baseline (Normalisasi)**
2. **Baseline + Smote**
3. **Baseline + Feature Selection**
4. **Baseline + Smote + Feature Selection**

**Kombinasikan semua Langkah diatas menggunakan metode normalisasi**

**Alur semua:**

**Normalisasi -> Smote -> Feature Selection -> Model**

**Metode Normalisasi:**

1. **Min Max Scaling**
2. **Z-Score (Standard Scaling)**
3. **Decimal Scaling**
4. **Robust Scaler**
5. **L2 Normalization**
6. **Log Transformation**

**Note:**

**Normalisasi dilakukan pada kedua dataset training dan testing**

**Ringkasan Akurasi Feature Selection**

Berikut adalah akurasi untuk setiap percobaan:

1. **Tanpa Pregnancies**: 0.785714
2. **Tanpa Glucose**: 0.688312
3. **Tanpa BloodPressure**: 0.772727
4. **Tanpa SkinThickness**: 0.792208
5. **Tanpa Insulin**: 0.766234
6. **Tanpa BMI**: 0.740260
7. **Tanpa DiabetesPedigreeFunction**: 0.746753
8. **Tanpa Age**: 0.772727

**Analisis**

Dari ringkasan di atas, kita dapat melihat bahwa akurasi tertinggi dicapai ketika fitur **Pregnancies** dihilangkan, dengan akurasi **0.785714**. Namun, fitur **SkinThickness** juga memberikan hasil yang baik dengan akurasi **0.792208**, yang merupakan akurasi tertinggi di antara semua percobaan.

**Kesimpulan**

* **Fitur yang memberikan akurasi terbaik**: **SkinThickness** (0.792208)
* **Fitur yang memberikan akurasi terendah**: **Glucose** (0.688312)

Dari analisis ini, Anda mungkin ingin mempertimbangkan untuk menggunakan model dengan fitur **SkinThickness** yang dihilangkan, karena memberikan akurasi terbaik. Namun, penting juga untuk mempertimbangkan metrik lain seperti precision, recall, dan f1-score, terutama jika ada ketidakseimbangan kelas dalam dataset Anda.

# Feature Selection - Uji Model dengan Setiap Fitur

Performance for feature 'Pregnancies':

Accuracy: 0.6038961038961039

Precision: 0.43478260869565216

Recall: 0.36363636363636365

F1-score: 0.39603960396039606

Performance for feature 'Glucose':

Accuracy: 0.7012987012987013

Precision: 0.5692307692307692

Recall: 0.6727272727272727

F1-score: 0.6166666666666667

Performance for feature 'BloodPressure':

Accuracy: 0.5714285714285714

Precision: 0.3103448275862069

Recall: 0.16363636363636364

F1-score: 0.21428571428571427

Performance for feature 'SkinThickness':

Accuracy: 0.577922077922078

Precision: 0.3611111111111111

Recall: 0.23636363636363636

F1-score: 0.2857142857142857

Performance for feature 'Insulin':

Accuracy: 0.5974025974025974

Precision: 0.41025641025641024

Recall: 0.2909090909090909

F1-score: 0.3404255319148936

Performance for feature 'BMI':

Accuracy: 0.6298701298701299

Precision: 0.4838709677419355

Recall: 0.5454545454545454

F1-score: 0.5128205128205128

Performance for feature 'DiabetesPedigreeFunction':

Accuracy: 0.538961038961039

Precision: 0.3787878787878788

Recall: 0.45454545454545453

F1-score: 0.4132231404958678

Performance for feature 'Age':

Accuracy: 0.577922077922078

Precision: 0.38095238095238093

Recall: 0.2909090909090909

F1-score: 0.32989690721649484

# Feature Selection - RFE dengan Permutation Importance

Feature: Glucose, Importance: 0.06948051948051948

Feature: BMI, Importance: 0.01753246753246751

Feature: DiabetesPedigreeFunction, Importance: 0.013636363636363624

Feature: Insulin, Importance: 0.007142857142857129

Feature: Age, Importance: -0.0006493506493506663

Feature: SkinThickness, Importance: -0.007792207792207795

Feature: Pregnancies, Importance: -0.011688311688311692

Feature: BloodPressure, Importance: -0.037662337662337654

'BloodPressure'

Accuracies for each K:

K = 1, Accuracy = 0.61

K = 3, Accuracy = 0.66

K = 5, Accuracy = 0.67

K = 7, Accuracy = 0.68

K = 9, Accuracy = 0.71

K = 11, Accuracy = 0.71

K = 13, Accuracy = 0.71

K = 15, Accuracy = 0.71

K = 17, Accuracy = 0.70

K = 19, Accuracy = 0.70

'BloodPressure', 'Age'

Accuracies for each K:

K = 1, Accuracy = 0.58

K = 3, Accuracy = 0.62

K = 5, Accuracy = 0.68

K = 7, Accuracy = 0.71

K = 9, Accuracy = 0.68

K = 11, Accuracy = 0.68

K = 13, Accuracy = 0.68

K = 15, Accuracy = 0.68

K = 17, Accuracy = 0.71

K = 19, Accuracy = 0.73

'BloodPressure', 'Age', 'Insulin'

Accuracies for each K:

K = 1, Accuracy = 0.56

K = 3, Accuracy = 0.62

K = 5, Accuracy = 0.68

K = 7, Accuracy = 0.69

K = 9, Accuracy = 0.68

K = 11, Accuracy = 0.68

K = 13, Accuracy = 0.68

K = 15, Accuracy = 0.68

K = 17, Accuracy = 0.68

K = 19, Accuracy = 0.69

'BloodPressure', 'Age', 'Insulin', 'SkinThickness'

Accuracies for each K:

K = 1, Accuracy = 0.64

K = 3, Accuracy = 0.70

K = 5, Accuracy = 0.69

K = 7, Accuracy = 0.69

K = 9, Accuracy = 0.71

K = 11, Accuracy = 0.72

K = 13, Accuracy = 0.73

K = 15, Accuracy = 0.73

K = 17, Accuracy = 0.72

K = 19, Accuracy = 0.73

'BloodPressure', 'Age', 'Insulin', 'SkinThickness', 'Pregnancies'

Accuracies for each K:

K = 1, Accuracy = 0.71

K = 3, Accuracy = 0.71

K = 5, Accuracy = 0.72

K = 7, Accuracy = 0.71

K = 9, Accuracy = 0.73

K = 11, Accuracy = 0.74

K = 13, Accuracy = 0.75

K = 15, Accuracy = 0.74

K = 17, Accuracy = 0.73

K = 19, Accuracy = 0.73

# Normalisasi

Kode yang Anda berikan **menggunakan fit\_transform() hanya pada data training**, dan menggunakan transform() pada data testing. Ini adalah praktik yang benar dalam pembelajaran mesin.

**Analisis Kode:**

1. **Pada Data Training (X\_train)**:

python

Copy code

X\_train\_scaled = pd.DataFrame(scaler.fit\_transform(X\_train), columns=X\_train.columns)

* + **fit\_transform()** digunakan pada data training.
  + fit\_transform() menghitung parameter normalisasi (nilai minimum dan maksimum dari setiap kolom) dan menerapkan normalisasi pada data training.
  + Parameter yang dihitung (min dan max) disimpan dalam objek scaler.

1. **Pada Data Testing (X\_test)**:

python

Copy code

X\_test\_scaled = pd.DataFrame(scaler.transform(X\_test), columns=X\_test.columns)

* + **transform()** digunakan pada data testing.
  + transform() hanya menerapkan normalisasi berdasarkan parameter (min dan max) yang dihitung dari data training (melalui fit\_transform()).
  + Tidak ada parameter baru yang dihitung dari data testing.

**Mengapa Ini Penting?**

* Dengan menggunakan **fit\_transform() hanya pada data training**, Anda memastikan bahwa model tidak "mengintip" data testing.
* **transform() pada data testing** memastikan bahwa skala yang diterapkan pada data testing konsisten dengan skala data training, yang mencerminkan skenario nyata di mana model dihadapkan pada data baru.

**Kesimpulan**

Kode tersebut hanya menggunakan **fit\_transform() pada data training** dan **transform() pada data testing**, sesuai dengan praktik terbaik. Hal ini memastikan konsistensi skala dan mencegah data leakage.

# Feature Selection

**Feature selection** sebaiknya dilakukan **sebelum melatih model**. Tujuannya adalah untuk memilih fitur-fitur yang relevan dan mengurangi dimensi dataset agar model lebih efisien, lebih cepat, dan lebih akurat. Berikut adalah alasan mengapa feature selection harus dilakukan sebelum melatih model:

**Alasan Melakukan Feature Selection Sebelum Melatih Model:**

1. **Menghindari Overfitting**:
   * Fitur yang tidak relevan atau berisik (noisy) dapat menyebabkan model menjadi terlalu kompleks dan rentan terhadap overfitting. Dengan memilih hanya fitur yang paling relevan, Anda dapat mengurangi risiko ini.
2. **Meningkatkan Kinerja Model**:
   * Mengurangi jumlah fitur dapat membantu model untuk berfokus hanya pada informasi yang paling penting. Hal ini dapat meningkatkan kecepatan pelatihan dan juga meningkatkan generalisasi model.
3. **Mengurangi Kompleksitas**:
   * Jika dataset memiliki terlalu banyak fitur, melatih model dengan semua fitur bisa memakan waktu yang sangat lama dan memerlukan lebih banyak sumber daya komputasi. Dengan feature selection, model akan lebih efisien.
4. **Memastikan Model Tidak Menggunakan Fitur yang Tidak Relevan**:
   * Fitur-fitur yang tidak relevan atau memiliki korelasi rendah dengan target variabel (misalnya, kelas dalam klasifikasi) tidak memberikan informasi yang berguna dan hanya memperumit model.

**Proses Umum:**

1. **Pilih fitur relevan menggunakan teknik feature selection** (seperti SelectKBest, Recursive Feature Elimination (RFE), atau teknik lainnya).
2. **Latih model** menggunakan data yang telah dipilih fiturnya.
3. **Evaluasi kinerja model** setelah training.

**Teknik Feature Selection Lainnya:**

Selain SelectKBest, ada beberapa teknik lain yang bisa digunakan untuk pemilihan fitur, antara lain:

* **Recursive Feature Elimination (RFE)**: Menghapus fitur yang paling tidak relevan secara berulang dan melatih model di setiap iterasi.
* **Feature Importance (misalnya menggunakan RandomForest)**: Menggunakan algoritma seperti Random Forest untuk mengevaluasi pentingnya setiap fitur.
* **PCA (Principal Component Analysis)**: Untuk reduksi dimensi, meskipun ini bukan pemilihan fitur langsung, tetapi lebih ke transformasi data.

**Kapan Feature Selection Bisa Dilakukan Setelah Pelatihan?**

Meskipun **umumnya dilakukan sebelum pelatihan**, ada situasi di mana Anda bisa mengevaluasi fitur setelah pelatihan, misalnya untuk mengevaluasi **feature importance** (seperti yang dilakukan dengan Random Forest atau XGBoost). Namun, **pemilihan fitur sebelum pelatihan** tetap merupakan pendekatan yang lebih baik dan lebih umum.

Jika Anda ingin mendiskusikan lebih lanjut tentang teknik feature selection lainnya atau ada yang ingin ditambahkan, beri tahu saya!

**Feature selection** sebaiknya dilakukan **setelah SMOTE**.

**Alasan mengapa feature selection dilakukan setelah SMOTE:**

1. **SMOTE menghasilkan data baru (synthetic)** yang akan membantu memastikan bahwa **fitur yang dipilih relevan dengan distribusi data yang baru**. Jika Anda melakukan feature selection sebelum SMOTE, fitur yang dipilih mungkin tidak mencakup aspek penting dari data sintetis yang dihasilkan oleh SMOTE, karena SMOTE memperkenalkan sampel baru berdasarkan data minoritas.
2. **SMOTE mempengaruhi distribusi data**, terutama pada kelas minoritas. Setelah melakukan SMOTE, distribusi data menjadi lebih seimbang, dan fitur yang dipilih seharusnya mempertimbangkan distribusi data yang baru tersebut. Memilih fitur sebelum SMOTE berisiko mengabaikan informasi penting yang relevan setelah resampling.

**Langkah yang Disarankan:**

1. **Lakukan SMOTE** untuk mengatasi ketidakseimbangan kelas dan menghasilkan data yang lebih seimbang.
2. **Lakukan feature selection** untuk memilih fitur terbaik berdasarkan data yang telah di-resample oleh SMOTE.
3. **Latih model** menggunakan data dengan fitur yang telah dipilih.

**Urutan Proses:**

1. **Resample dengan SMOTE** (membuat data seimbang).
2. **Pilih fitur** yang relevan setelah SMOTE.
3. **Latih model** menggunakan data dengan fitur terpilih.

**Kenapa Begitu?**

* **SMOTE membuat data seimbang**, sehingga distribusi kelas dalam dataset menjadi lebih representatif untuk melatih model yang baik.
* Setelah SMOTE, Anda bisa memilih fitur yang lebih relevan berdasarkan distribusi data baru yang sudah seimbang. Ini akan menghindari pemilihan fitur yang mungkin hanya relevan pada dataset yang sangat tidak seimbang, atau memilih fitur yang kurang menggambarkan data minoritas.

**Kesimpulan:**

Feature selection **setelah SMOTE** memberikan keuntungan karena model yang dilatih menggunakan data sintetis akan lebih baik mencerminkan distribusi data yang baru dan fitur yang relevan untuk data tersebut. Jika Anda melakukan feature selection sebelum SMOTE, fitur yang dipilih mungkin tidak sepenuhnya mencerminkan distribusi kelas minoritas yang telah diperbaiki oleh SMOTE.