MODUL 5. Feature Selection (Seleksi Fitur)

Apa itu Feature Selection?

Feature Selection (seleksi fitur) adalah proses pemilihan subset dari fitur yang paling relevan dari kumpulan data untuk digunakan dalam membangun model prediktif. Dalam pengolahan data, seringkali terdapat banyak fitur yang mungkin tidak semuanya relevan, dan pemilihan fitur yang tepat dapat membantu:

1. **Mengurangi Dimensi**: Mengurangi jumlah fitur untuk mengurangi kompleksitas model.
2. **Meningkatkan Akurasi**: Dengan hanya menggunakan fitur yang relevan, kita dapat meningkatkan performa model.
3. **Mengurangi Overfitting**: Menghilangkan fitur yang tidak relevan atau berlebihan dapat mengurangi risiko overfitting.
4. **Meningkatkan Kecepatan Komputasi**: Model dengan lebih sedikit fitur membutuhkan lebih sedikit waktu untuk dilatih dan diprediksi.
5. **Meningkatkan Interpretabilitas**: Dengan memilih fitur yang benar-benar penting, hasil model lebih mudah dipahami.

Tujuan Feature Selection

Tujuan utama dari feature selection adalah menemukan subset fitur yang paling relevan untuk model dengan cara:

1. **Menghilangkan Fitur Redundan**: Fitur yang memberikan informasi yang sama atau sangat mirip dengan fitur lain.
2. **Menghilangkan Fitur yang Tidak Relevan**: Fitur yang tidak memiliki hubungan atau sangat lemah dengan target variabel.
3. **Meningkatkan Performa Model**: Dengan memilih fitur yang penting, kita dapat meningkatkan kemampuan generalisasi model.

Metode Feature Selection

> Terdapat beberapa metode dalam Feature Selection yang umum digunakan, termasuk Selection Forward, Selection Backward, Bidirectional Selection, dan Random Generation. Berikut penjelasan setiap metode:

1. Selection Forward (Seleksi Maju)

> Seleksi maju (Selection Forward) adalah metode seleksi fitur di mana kita memulai tanpa fitur apa pun, kemudian menambahkan fitur satu per satu ke dalam model berdasarkan kontribusi mereka terhadap peningkatan kinerja model. Pada setiap iterasi, fitur yang paling signifikan ditambahkan ke dalam model sampai tidak ada lagi fitur yang memberikan peningkatan signifikan pada kinerja.

Langkah-langkah:

1. Mulai dengan model tanpa fitur.
2. Evaluasi kinerja model dengan menambahkan satu fitur ke dalam model.
3. Tambahkan fitur yang memberikan peningkatan kinerja terbesar.
4. Ulangi proses ini sampai tidak ada lagi peningkatan signifikan.

Keuntungan:

1. Sederhana dan mudah diimplementasikan.
2. Memberikan hasil yang baik dalam menemukan fitur yang paling relevan.

Kerugian:

1. Proses ini bisa memakan waktu untuk dataset dengan banyak fitur, karena setiap fitur perlu diuji satu per satu.

Pseudocode

```
#
Set Selected_Features = {}  // Subset fitur terpilih
Set Best_Score = 0

WHILE true:
    For each feature F not in Selected_Features:
        Add F to Selected_Features
        Train model on Selected_Features
        Evaluate performance Score
        If Score > Best_Score:
            Best_Feature = F
            Best_Score = Score
    IF no improvement in Best_Score:
        BREAK
    ELSE:
        Add Best_Feature to Selected_Features
```

Contoh code untuk metode seleksi maju

```
#
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import accuracy_score

penguins = pd.read_csv('penguins.csv')
# Drop rows with missing values
penguins = penguins.dropna()

X = penguins[['bill_length', 'bill_depth', 'flipper_length', 'body_mass']]
y = penguins['species']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model yang digunakan
model = RandomForestClassifier(random_state=42)

# Seleksi Maju (Forward Selection)
sfs = SequentialFeatureSelector(model, n_features_to_select=2, direction='forward')
sfs.fit(X_train, y_train)

print("Fitur yang dipilih:", sfs.get_support(indices=True))

X_train_selected = sfs.transform(X_train)
X_test_selected = sfs.transform(X_test)
model.fit(X_train_selected, y_train)
y_pred = model.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi model dengan fitur yang dipilih:", accuracy)
```

2. Selection Backward (Seleksi Mundur)

> Seleksi mundur (Selection Backward) adalah kebalikan dari seleksi maju, di mana kita memulai dengan semua fitur yang tersedia, kemudian secara bertahap menghapus fitur yang paling tidak berpengaruh terhadap performa model. Proses ini terus berlanjut sampai hanya tersisa fitur yang paling signifikan.

Langkah-langkah:

1. Mulai dengan model yang menggunakan semua fitur.
2. Evaluasi kinerja model dengan menghapus satu fitur setiap kali.
3. Hapus fitur yang kontribusinya paling sedikit terhadap kinerja model.
4. Ulangi proses ini sampai tidak ada fitur lagi yang dapat dihapus tanpa menurunkan kinerja secara signifikan.

Keuntungan:

1. Cocok untuk dataset dengan jumlah fitur yang relatif sedikit.
2. Mengidentifikasi fitur yang benar-benar penting dengan cepat.

Kerugian: Lambat untuk dataset dengan jumlah fitur yang besar, karena harus mengevaluasi banyak fitur.

Pseudocode

```
#
Set Selected_Features = All_Features
Set Best_Score = Evaluate model on All_Features

WHILE Selected_Features is not empty:
    For each feature F in Selected_Features:
        Remove F from Selected_Features
        Train model on remaining features
        Evaluate performance Score
        If Score > Best_Score:
            Best_Score = Score
            Worst_Feature = F
    IF no improvement or all features are necessary:
        BREAK
    ELSE:
        Remove Worst_Feature from Selected_Features
```

Contoh Code Seleksi Mundur

```
#
from sklearn.feature_selection import SequentialFeatureSelector

# Seleksi Mundur (Backward Selection)
sbs = SequentialFeatureSelector(model, n_features_to_select=2, direction='backward')
```

```
sbs.fit(X, y)

# Fitur yang dipilih
print("Fitur yang dipilih:", sbs.get_support(indices=True))

X_selected = sbs.transform(X)
model.fit(X_selected, y)
```

3. **Bidirectional Selection (Seleksi Dua Arah)**

> Seleksi dua arah (Bidirectional Selection) menggabungkan pendekatan seleksi maju dan seleksi mundur. Pada setiap iterasi, baik fitur dapat ditambahkan (seperti pada seleksi maju) atau dihapus (seperti pada seleksi mundur) tergantung pada pengaruhnya terhadap kinerja model.

Langkah-langkah:

1. Mulai dengan model kosong atau dengan semua fitur.
2. Evaluasi fitur yang dapat ditambahkan atau dihapus pada setiap iterasi.
3. Pilih untuk menambahkan atau menghapus fitur berdasarkan peningkatan atau penurunan kinerja model.
4. Ulangi sampai model mencapai kinerja terbaik dengan fitur optimal.

Keuntungan:

1. Memberikan hasil yang lebih optimal dengan mempertimbangkan penambahan dan penghapusan fitur secara dinamis.

Kerugian:

1. Memerlukan lebih banyak waktu dan sumber daya dibandingkan dengan seleksi maju atau mundur saja.

Pseudocode

```
#
Set Selected_Features = All_Features  // Semua fitur dimulai dalam subset
Set Best_Score = Evaluate model on All_Features

WHILE Selected_Features is not empty:
    For each feature F in Selected_Features:
        Remove F from Selected_Features
        Train model on remaining features
```

```
        Evaluate performance Score
        If Score > Best_Score:
            Best_Score = Score
            Worst_Feature = F
    IF no improvement or all features are necessary:
        BREAK
    ELSE:
        Remove Worst_Feature from Selected_Features
```

Contoh Code Seleksi dua arah

```
#
sfs_bidirectional = SequentialFeatureSelector(model, n_features_to_select=2,
                    direction='forward', tol=1e-4)
sfs_bidirectional.fit(X, y)

# Fitur yang dipilih
print("Fitur yang dipilih:", sfs_bidirectional.get_support(indices=True))

# Evaluasi model dengan fitur yang dipilih
X_selected = sfs_bidirectional.transform(X)
model.fit(X_selected, y)
```

4. **Random Generation (Pembuatan Acak)**

> Random Generation adalah metode eksploratif di mana fitur dipilih secara acak dari kumpulan fitur yang tersedia, kemudian model dibangun dan dievaluasi berdasarkan kombinasi fitur acak tersebut. Metode ini digunakan untuk menemukan kombinasi fitur yang tidak biasa namun dapat memberikan kinerja yang baik

Langkah-langkah:

1. Pilih subset fitur secara acak dari seluruh kumpulan fitur.
2. Bangun model berdasarkan subset fitur acak tersebut.
3. Evaluasi kinerja model.
4. Simpan kombinasi fitur dengan kinerja terbaik.

Keuntungan:

1. Dapat menemukan kombinasi fitur yang tidak terduga yang memberikan kinerja yang baik.

Kerugian:

1. Tidak efisien, terutama untuk dataset besar dengan banyak fitur, karena memerlukan banyak percobaan acak

Pseudocode

```
#
Set Selected_Features = All_Features  // Semua fitur dimulai dalam subset
Set Best_Score = Evaluate model on All_Features

WHILE Selected_Features is not empty:
    For each feature F in Selected_Features:
        Remove F from Selected_Features
        Train model on remaining features
        Evaluate performance Score
        If Score > Best_Score:
            Best_Score = Score
            Worst_Feature = F
    IF no improvement or all features are necessary:
        BREAK
    ELSE:
        Remove Worst_Feature from Selected_Features
```

Contoh Code Untuk Metode Random Generation

```
#
import numpy as np
from itertools import combinations
from sklearn.metrics import accuracy_score

# Pilih fitur secara acak
n_features = X.shape[1]
all_combinations = list(combinations(range(n_features), 2))  # Kombinasi 2 fitur

best_accuracy = 0
best_combination = None

for comb in all_combinations:
    # Use .iloc to select columns by integer index
    X_selected = X.iloc[:, list(comb)]
    model.fit(X_selected, y)
    y_pred = model.predict(X_selected)
    acc = accuracy_score(y, y_pred)
```

```
    if acc > best_accuracy:
        best_accuracy = acc
        best_combination = comb

print(f"Kombinasi terbaik: {best_combination} dengan akurasi {best_accuracy}")
```

Latihan

> Dataset dapat di download pada link [teks link]https://drive.google.com/drive/folders/1Qr_eeUPSTpiuowvvUrn4VN00UsSeib3T?usp=sharing()

```
#
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SequentialFeatureSelector
from itertools import combinations

# Load dataset segmentation-all.csv ( pastikan file tersebut ada dalam direktori yang sama )
data = pd.read_csv('segmentation-all.csv')

# Pisahkan fitur dan target
X = data.drop('Class', axis=1)
y = data['Class']

# Bagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model yang digunakan
model = RandomForestClassifier(random_state=42)


# Seleksi Maju (Forward Selection)
sfs = SequentialFeatureSelector(model, n_features_to_select=5, direction='forward')
sfs.fit(X_train, y_train)

# Fitur yang dipilih
```

```
print("Fitur yang dipilih (Forward Selection):", sfs.get_support(indices=True))

# Evaluasi model dengan fitur yang dipilih
X_train_selected = sfs.transform(X_train)
X_test_selected = sfs.transform(X_test)
model.fit(X_train_selected, y_train)
y_pred = model.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi model dengan fitur yang dipilih (Forward Selection):", accuracy)



# Seleksi Mundur (Backward Selection)
sbs = SequentialFeatureSelector(model, n_features_to_select=5, direction='backward')
sbs.fit(X_train, y_train)

# Fitur yang dipilih
print("\nFitur yang dipilih (Backward Selection):", sbs.get_support(indices=True))

# Evaluasi model dengan fitur yang dipilih
X_train_selected = sbs.transform(X_train)
X_test_selected = sbs.transform(X_test)
model.fit(X_train_selected, y_train)
y_pred = model.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi model dengan fitur yang dipilih (Backward Selection):", accuracy)




# Seleksi dua arah (Bidirectional Selection)
sfs_bidirectional = SequentialFeatureSelector(model, n_features_to_select=5, direction='forward')
sfs_bidirectional.fit(X_train, y_train)

# Fitur yang dipilih
print("\nFitur yang dipilih (Bidirectional Selection):", sfs_bidirectional.get_support(indices=True))

# Evaluasi model dengan fitur yang dipilih
X_train_selected = sfs_bidirectional.transform(X_train)
X_test_selected = sfs_bidirectional.transform(X_test)
model.fit(X_train_selected, y_train)
y_pred = model.predict(X_test_selected)
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi model dengan fitur yang dipilih (Bidirectional Selection):", accuracy)
```

```
# Random Generation
n_features = X.shape[1]
all_combinations = list(combinations(range(n_features), 5))  # Kombinasi 5 fitur

best_accuracy = 0
best_combination = None

for comb in all_combinations:
    # Use .iloc to select columns by integer index
    X_selected = X.iloc[:, list(comb)]
    X_train_selected, X_test_selected, y_train, y_test = train_test_split(X_selected, y, test_size=0.
    model.fit(X_train_selected, y_train)
    y_pred = model.predict(X_test_selected)
    acc = accuracy_score(y_test, y_pred)

    if acc > best_accuracy:
        best_accuracy = acc
        best_combination = comb

print("\nKombinasi terbaik (Random Generation):", best_combination, "dengan akurasi", best_accuracy)
```