PRAKTIKUM MACHINE LEARNING

UNIT 7 : BUILDING MODEL (BAGIAN II)

adult_klasifikasi.ipynb

Eka Praja Wiyata Mandala, S.Kom, M,Kom, CADS

Unit 7 : Building Model

Cek Underfitting atau Overfitting untuk Hyperparameter

Underfitting

terjadi ketika model terlalu sederhana untuk menangkap pola yang kompleks dalam data. Model ini tidak mampu memetakan hubungan yang sebenarnya antara fitur dan target variabel

Ciri-ciri:

- Akurasi rendah baik pada data pelatihan maupun data uji.
- Model terlalu umum dan tidak spesifik untuk data yang sedang dipelajari.
- Bias tinggi.

Unit 7 : Building Model

Penyebab:

- Model yang terlalu sederhana (misalnya, jumlah pohon keputusan yang terlalu sedikit).
- Data pelatihan yang terlalu sedikit atau tidak representatif.
- Hyperparameter yang terlalu membatasi kompleksitas model.

Contoh: Anda mencoba melatih model untuk memprediksi harga rumah berdasarkan luas dan lokasi. Namun, model hanya mempertimbangkan luas saja, sehingga tidak dapat menangkap pengaruh lokasi terhadap harga.

Unit 7 : Building Model

Overfitting

terjadi ketika model terlalu kompleks dan menghafal noise atau variasi acak dalam data pelatihan. Akibatnya, model menjadi terlalu spesifik untuk data pelatihan dan tidak dapat generalisasi dengan baik pada data baru.

Ciri-ciri:

- Akurasi tinggi pada data pelatihan, tetapi rendah pada data uji.
- Model terlalu kompleks dan rentan terhadap noise.
- Variansi tinggi.

Penyebab:

- Model yang terlalu kompleks (misalnya, jumlah pohon keputusan yang terlalu banyak).
- Data pelatihan yang terlalu sedikit atau mengandung noise.
- Hyperparameter yang terlalu memungkinkan kompleksitas model.

Contoh: Anda melatih model untuk mengenali tulisan tangan. Model berhasil mengenali tulisan tangan dari penulis tertentu dalam data pelatihan, tetapi gagal mengenali tulisan tangan dari penulis lain.

Unit 7 : Building Model

Hubungan dengan Hyperparameter Tuning

Hyperparameter tuning bertujuan untuk **menemukan nilai optimal dari hyperparameter** agar model dapat generalisasi dengan baik pada data baru. Jika **nilai hyperparameter terlalu kecil**, model akan mengalami **underfitting**. Sebaliknya, jika **nilai hyperparameter terlalu besar**, model akan mengalami **overfitting**.

Tujuan utama hyperparameter tuning adalah:

- **Mencegah overfitting**: Dengan memilih hyperparameter yang tepat, kita dapat membatasi kompleksitas model dan mencegahnya menghafal noise dalam data.
- **Mencegah underfitting**: Dengan memilih hyperparameter yang cukup kompleks, kita dapat memastikan model mampu menangkap pola yang kompleks dalam data.

Unit 7 : Building Model

Membandingkan Akurasi Training vs Testing:

```
[ ]  from sklearn.metrics import accuracy_score

     # Untuk model Decision Tree terbaik
     y_train_pred = best_dt_model.predict(X_train)
     y_test_pred = best_dt_model.predict(X_test)

     train_accuracy = accuracy_score(y_train, y_train_pred)
     test_accuracy = accuracy_score(y_test, y_test_pred)

     print("Training Accuracy:", train_accuracy)
     print("Testing Accuracy:", test_accuracy)
```

Interpretasi:

- Jika **train_accuracy jauh lebih tinggi dari test_accuracy**, ini indikasi **overfitting**.
- Jika **keduanya rendah dan hampir sama**, ini bisa jadi indikasi **underfitting**.

Unit 7 : Building Model

Learning curve

```
from sklearn.model_selection import learning_curve
import matplotlib.pyplot as plt

train_sizes, train_scores, test_scores = learning_curve(
    best_dt_model, X_train, y_train, cv=5,
    train_sizes=np.linspace(0.1, 1.0, 10), n_jobs=-1)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.figure(figsize=(10,6))
plt.plot(train_sizes, train_mean, label='Training score')
plt.plot(train_sizes, test_mean, label='Cross-validation score')
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std,
alpha=0.1)
```

Unit 7 : Building Model


```
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std,
alpha=0.1)
plt.xlabel('Training Set Size')
plt.ylabel('Accuracy Score')
plt.title('Learning Curve')
plt.legend()
plt.show()
```

Interpretasi:

- Jika **kurva training terus naik tapi kurva testing mendatar atau turun**, ini indikasi **overfitting**.
- Jika k**edua kurva rendah dan berdekatan**, ini indikasi **underfitting**.

Unit 7 : Building Model

Cross Validation Score

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(best_dt_model, X_train, y_train, cv=5)
print("Cross-validation scores:", cv_scores)
print("Mean CV score:", np.mean(cv_scores))
print("Standard deviation of CV score:", np.std(cv_scores))
```

Interpretasi:

- Jika **mean CV score jauh lebih rendah dari training accuracy**, ini bisa jadi indikasi **overfitting**.
- Jika **standard deviasi CV score tinggi**, ini bisa menunjukkan **model yang tidak stabil**, yang juga bisa mengindikasikan **overfitting**.

Unit 7 : Building Model

Complexity Curve (untuk Decision Tree):

```
max_depths = range(1, 21)  #digunakan untuk nilai integer dari 1 sampai 20
train_scores = []
test_scores = []

for max_depth in max_depths:
    dt = DecisionTreeClassifier(max_depth=max_depth, random_state=42)
    dt.fit(X_train, y_train)
    train_scores.append(accuracy_score(y_train, dt.predict(X_train)))
    test_scores.append(accuracy_score(y_test, dt.predict(X_test)))

plt.figure(figsize=(10,6))
plt.plot(max_depths, train_scores, label='Training score')
plt.plot(max_depths, test_scores, label='Testing score')
plt.xlabel('Max Depth')
plt.ylabel('Accuracy Score')
plt.title('Model Complexity Curve')
plt.legend()
```

Unit 7 : Building Model

Interpretasi:

- Jika **kurva training terus naik tapi kurva testing mulai turun setelah titik tertentu**, ini indikasi **overfitting**.
- Jika **kedua kurva rendah dan berdekatan**, ini indikasi **underfitting**.

observasi:

- **Underfitting**: Terlihat di bagian kiri grafik (kedalaman 1-3) dimana kedua skor rendah.
- **Overfitting**: Mulai terjadi setelah kedalaman sekitar 7, dimana garis biru terus naik tapi garis oranye mulai turun.
- **Sweet spot (titik optimal)**: Berada di sekitar kedalaman 5-7, dimana testing score mencapai puncak.

Kesimpulan:

- Model dengan **kedalaman sekitar 5-7 mungkin memberikan keseimbangan terbaik antara bias dan varians**.
- Setelah kedalaman 7, model mulai overfitting: performa pada data training terus meningkat, tapi menurun pada data testing.
- Kedalaman pohon di atas 10 menunjukkan overfitting yang signifikan, dengan gap besar antara training dan testing score.

Rekomendasi:

- Gunakan **max_depth antara 5-7** untuk model final untuk **menghindari overfitting**.

Unit 7 : Building Model

Visualisasi Decision Tree versi Standar dengan max_depth=5

```
from sklearn.tree import plot_tree
```

```
# Buat gambar dengan ukuran yang sangat besar - disini maxdepth dibuat 5 agar
lebih cepat terbuat
plt.figure(figsize=(100, 50))  # Ukuran dalam inci

# Plot pohon keputusan
plot_tree(dt_model, feature_names=X.columns, class_names=['<=50K', '>50K'],
          filled=True, rounded=True, fontsize=10, max_depth=5)

# Simpan gambar dengan DPI tinggi
plt.savefig(f'{folder_name}/decision_tree_visualization.png', dpi=300,
bbox_inches='tight')

print("Gambar pohon keputusan telah disimpan sebagai
'decision_tree_visualization.png'")
```

Unit 7 : Building Model

Visualisasi Decision Tree versi Hyperparameter tanpa max_depth

```
# Buat gambar dengan ukuran yang sangat besar
plt.figure(figsize=(100, 50))  # Ukuran dalam inci

# Plot pohon keputusan
plot_tree(best_dt_model, feature_names=X.columns, class_names=['<=50K', '>50K'],
          filled=True, rounded=True, fontsize=10)

# Simpan gambar dengan DPI tinggi
plt.savefig(f'{folder_name}/best_decision_tree_visualization.png', dpi=300,
bbox_inches='tight')

print("Gambar pohon keputusan telah disimpan sebagai
'best_decision_tree_visualization.png'")
```

Unit 7 : Building Model

Visualisasi Decision Tree versi Hyperparameter dengan max_depth=5

```
# Buat gambar dengan ukuran yang sangat besar
plt.figure(figsize=(100, 50))  # Ukuran dalam inci

# Plot pohon keputusan
plot_tree(best_dt_model, feature_names=X.columns, class_names=['<=50K', '>50K'],
          filled=True, rounded=True, fontsize=10, max_depth=5)

# Simpan gambar dengan DPI tinggi
plt.savefig(f'{folder_name}/best_decision_tree_visualization.png', dpi=300,
bbox_inches='tight')

print("Gambar pohon keputusan telah disimpan sebagai
'best_decision_tree_visualization.png'")
```

Semua Unit Selesai