Nama : Michael Christopher
NIM : 1103210260
Analisis UTS ML Soal #1

Analisis Mendalam Kasus Machine Learning

1. Strategi Mengatasi Underfitting pada Linear Regression dan Decision Tree

Underfitting terjadi ketika model terlalu sederhana sehingga tidak dapat menangkap kompleksitas pola dalam data, menyebabkan performa yang buruk baik pada data training maupun testing. Berikut strategi untuk meningkatkan performa model yang mengalami underfitting:

Pendekatan 1: Transformasi dan Penambahan Fitur

Untuk Linear Regression: Transformasi fitur dan feature engineering dapat secara signifikan meningkatkan kapasitas model linear. Secara teknis, linear regression hanya dapat memodelkan hubungan linear, tetapi dengan transformasi yang tepat, kita bisa memodelkan hubungan non-linear.

- **Polynomial Features**: Menambahkan fitur kuadrat, kubik, dan interaksi antar fitur memungkinkan model linear menangkap hubungan non-linear. Misalnya, jika kita memiliki fitur X_1 dan X_2, kita dapat menambahkan X_1^2, X_2^2, dan X_1X_2.
- **Non-linear Transformasi**: Transformasi logaritmik, akar kuadrat, atau eksponensial pada fitur bisa membantu model linear menangkap hubungan non-linear yang spesifik.

Untuk Decision Tree:

- **Feature Engineering**: Membuat fitur baru yang lebih informatif dan representatif dari domain masalah, seperti rasio antar fitur atau fitur agregat.

Dampak pada Bias-Variance Tradeoff:

- Transformasi fitur mengurangi bias model dengan meningkatkan kompleksitas model, tetapi juga meningkatkan variance.
- Untuk dataset besar, peningkatan variance mungkin tidak menjadi masalah signifikan.
- Perlu hati-hati agar tidak terlalu banyak menambah fitur yang dapat menyebabkan overfitting. Teknik regularisasi mungkin diperlukan jika terlalu banyak fitur ditambahkan.

Pendekatan 2: Pengubahan Model ke Algoritma yang Lebih Kompleks

Untuk Linear Regression:

- **Beralih ke Model Non-linear**: Menggunakan model yang secara inheren non-linear seperti Support Vector Regression (SVR) dengan kernel non-linear, Neural Networks, atau Gaussian Process Regression.

Untuk Decision Tree:

- **Ensemble Methods**: Menggunakan Random Forest, Gradient Boosting (XGBoost, LightGBM), atau AdaBoost yang menggabungkan banyak decision tree untuk meningkatkan kapasitas dan stabilitas model.
- **Mengoptimalkan Hyperparameter**: Meningkatkan kedalaman maksimum tree, mengurangi minimum samples di leaf nodes, atau mengubah splitting criterion.

Dampak pada Bias-Variance Tradeoff:

- Model yang lebih kompleks seperti ensemble atau deep learning secara drastis mengurangi bias dengan meningkatkan kapasitas model.
- Namun, model ini juga lebih rentan terhadap overfitting (variance tinggi), terutama pada dataset kecil.
- Regularisasi dan early stopping menjadi lebih penting saat menggunakan model yang lebih kompleks.
- Ensemble methods seperti Random Forest cenderung memberikan keseimbangan yang baik antara bias dan variance karena efek rata-rata dari banyak model.

Perbandingan Kedua Pendekatan:

Transformasi Fitur vs. Model Kompleks:

- **Interpretabilitas**: Transformasi fitur pada linear regression tetap menghasilkan model yang relatif interpretatif, sementara model kompleks seperti ensemble atau neural networks cenderung menjadi "black box".
- **Skalabilitas**: Model kompleks mungkin memerlukan lebih banyak komputasi dan memori, sementara transformasi fitur lebih ringan secara komputasional.
- **Fleksibilitas**: Model kompleks biasanya lebih fleksibel dalam menangkap berbagai jenis pola, sementara transformasi fitur memerlukan pemahaman domain untuk memilih transformasi yang tepat.
- **Maintenance**: Model dengan transformasi fitur mungkin lebih mudah untuk dipelihara dan diperbarui dibandingkan model yang lebih kompleks.

Pilihan optimal bergantung pada karakteristik data, ukuran dataset, dan kebutuhan interpretabilitas. Pendekatan yang sering efektif adalah mencoba transformasi fitur terlebih dahulu, lalu beralih ke model yang lebih kompleks jika performa masih belum memuaskan.

2. Alternatif Loss Function untuk Masalah Regresi

Mean Squared Error (MSE) adalah loss function yang umum digunakan dalam masalah regresi, namun terdapat alternatif yang lebih sesuai untuk kasus-kasus tertentu:

Mean Absolute Error (MAE)

Definisi Matematis: $MAE = (1/n) \Sigma|y_i - \hat{y}_i|$

Keunggulan:

- **Robust terhadap outlier**: Karena menggunakan nilai absolut, bukan kuadrat, sehingga tidak memberikan penalti yang sangat besar pada error yang besar.
- **Interpretasi langsung**: Satuan MAE sama dengan satuan variabel target, menjadikannya lebih mudah diinterpretasi.
- **Median predictor**: Secara teoritis, model yang dioptimalkan dengan MAE akan memprediksi nilai median dari distribusi bersyarat.

Kelemahan:

- **Derivatif tidak kontinyu**: Pada titik error = 0, derivatif tidak terdefinisi, yang dapat menyulitkan algoritma gradient-based.
- **Lambat konvergen**: Algoritma optimasi untuk MAE dapat lebih lambat konvergen dibandingkan MSE.
- **Tidak sensitive terhadap arah**: MAE memberikan penalti yang sama untuk error positif dan negatif dengan magnitude yang sama.

Skenario optimal:

- Data dengan outlier signifikan
- Kasus di mana interpretasi langsung dari error sangat penting
- Aplikasi di mana median adalah statistik yang lebih representatif daripada mean (misalnya distribusi asimetris)
- Model prediksi harga rumah atau pendapatan yang sering memiliki distribusi skewed

Huber Loss

Definisi Matematis:

- Untuk $|y_i - \hat{y}_i| \leq \delta$: $L(y, \hat{y}) = 0.5 * (y - \hat{y})^2$
- Untuk $|y_i - \hat{y}_i| > \delta$: $L(y, \hat{y}) = \delta * (|y - \hat{y}| - 0.5 * \delta)$

di mana δ adalah parameter yang dapat disesuaikan.

Keunggulan:

- **Kombinasi terbaik**: Menggabungkan kelebihan MSE (differentiability) dan MAE (robustness to outliers).
- **Adaptif**: Parameter δ dapat disesuaikan berdasarkan tingkat outlier dalam data.
- **Diferensiabel**: Diferensiabel di semua titik, memudahkan optimasi gradient-based.

Kelemahan:

- **Hyperparameter tambahan**: Memerlukan tuning parameter δ, yang dapat menambah kompleksitas.
- **Kurang intuitif**: Formula yang lebih kompleks menjadikannya kurang intuitif dibandingkan MSE atau MAE.
- **Implementasi lebih rumit**: Tidak selalu tersedia secara default di semua library.

Skenario optimal:

- Dataset dengan outlier moderat
- Kasus di mana kita ingin keseimbangan antara robustness dan efisiensi komputasi
- Ketika kita memiliki pengetahuan prior tentang tingkat noise dalam data
- Aplikasi finansial atau peramalan ekonomi di mana outlier umum terjadi tetapi tidak ekstrem

Perbandingan Mendetail:

Sensitivitas terhadap Outlier: MSE > Huber > MAE (MSE paling sensitif)

Konvergensi dan Stabilitas Training: MSE > Huber > MAE (MSE biasanya konvergen paling cepat)

Kompleksitas Implementasi: MAE ≈ MSE < Huber (Huber lebih kompleks)

Kebutuhan Tuning Parameter: MAE = MSE < Huber (Huber membutuhkan parameter tambahan)

Kesesuaian dengan Distribusi Target:

- MSE: Optimal untuk distribusi Gaussian
- MAE: Optimal untuk distribusi Laplacian
- Huber: Optimal untuk distribusi dengan ekor tebal moderat

Pemilihan loss function yang tepat harus didasarkan pada karakteristik data, tujuan bisnis, dan pertimbangan komputasional. Pendekatan terbaik sering kali adalah mencoba beberapa loss function dan mengevaluasi performa model menggunakan validasi silang.

3. Metode Pengukuran Pentingnya Fitur Tanpa Mengetahui Nama Fitur

Bahkan tanpa mengetahui nama fitur, kita dapat menggunakan berbagai metode untuk mengukur pentingnya setiap fitur dalam model:

1. Permutation Feature Importance

Prinsip Teknikal: Metode ini bekerja dengan mengacak (permute) nilai fitur tertentu dan mengukur seberapa besar penurunan performa model akibat pengacakan tersebut. Fitur yang lebih penting akan menyebabkan penurunan performa yang lebih besar ketika diacak.

Langkah-langkah:

1. Hitung baseline performance model pada data validasi/testing.
2. Untuk setiap fitur:
    - Acak nilai fitur tersebut (memutus hubungan dengan target tanpa mengubah distribusi fitur).
    - Prediksi dengan model dan hitung performance baru.
    - Feature importance = baseline performance - new performance.
3. Urutkan fitur berdasarkan besarnya penurunan performa.

Keunggulan:

- **Model-agnostic**: Dapat digunakan pada berbagai jenis model, tidak hanya model tertentu.
- **Intuisi langsung**: Merepresentasikan dampak langsung fitur terhadap performa prediksi.
- **Tidak terpengaruh oleh collinearity**: Mengatasi kelemahan metode yang berbasis koefisien.

Keterbatasan:

- **Komputasi intensif**: Memerlukan banyak perhitungan, terutama untuk dataset besar.
- **Variabilitas**: Hasilnya dapat bervariasi antar running karena sifat stokastik dari pengacakan.
- **Hanya memperhitungkan efek marginal**: Mungkin tidak menangkap dengan baik interaksi kompleks antar fitur.

2. SHAP (SHapley Additive exPlanations) Values

Prinsip Teknikal: SHAP values berasal dari teori permainan kooperatif dan menghitung kontribusi marginal setiap fitur terhadap prediksi individual. Metode ini menghitung nilai Shapley, yang merepresentasikan rata-rata kontribusi marginal fitur di semua kemungkinan kombinasi fitur.

Langkah-langkah:

1. Untuk setiap prediksi dan setiap fitur, hitung nilai SHAP.
2. Nilai SHAP merepresentasikan seberapa besar fitur tersebut mengubah prediksi dari baseline (rata-rata prediksi).
3. Rata-rata nilai absolut SHAP untuk setiap fitur di seluruh dataset memberikan ukuran kepentingan global.

Keunggulan:

- **Keadilan teoritis**: Memenuhi properti keadilan dalam teori permainan (efficiency, symmetry, dummy, additivity).
- **Local dan global**: Menyediakan interpretasi untuk prediksi individual dan kepentingan fitur global.
- **Konsisten**: Berdasarkan framework matematis yang solid.

Keterbatasan:

- **Sangat intensif komputasi**: Terutama untuk model kompleks dan dataset besar.
- **Asumsi independensi**: Asumsi dasar SHAP mengasumsikan independensi fitur, yang sering tidak terpenuhi.
- **Interpretasi kompleks**: Memahami nilai Shapley memerlukan pemahaman konsep teoritis.

3. Coefficient-Based Methods (untuk Model Linear)

Prinsip Teknikal: Untuk model linear seperti linear regression, koefisien terstandarisasi dapat digunakan sebagai ukuran kepentingan fitur. Standarisasi fitur (mean=0, std=1) sebelum pelatihan memastikan koefisien dapat dibandingkan secara langsung.

Langkah-langkah:

1. Standarisasi semua fitur.
2. Latih model linear.
3. Gunakan nilai absolut dari koefisien sebagai ukuran kepentingan fitur.

Keunggulan:

- **Sederhana dan efisien**: Tidak memerlukan perhitungan tambahan selain pelatihan model.
- **Interpretasi langsung**: Koefisien menunjukkan hubungan langsung dengan target.
- **Statistik inferensial**: Dapat diperoleh p-values untuk menilai signifikansi statistik.

Keterbatasan:

- **Hanya untuk model linear**: Tidak dapat diterapkan pada model non-linear.
- **Dipengaruhi oleh multicollinearity**: Koefisien dapat tidak stabil jika terdapat korelasi tinggi antar fitur.
- **Tidak menangkap interaksi**: Mengasumsikan efek fitur adalah aditif dan independen.

Perbandingan Metode:

Untuk Dataset Kompleks dengan Interaksi Tinggi: SHAP > Permutation > Coefficient-based

Untuk Interpretabilitas dan Komunikasi: Coefficient-based > Permutation > SHAP

Untuk Efisiensi Komputasional: Coefficient-based > Permutation > SHAP

Untuk Model Non-linear Kompleks: SHAP > Permutation > Coefficient-based (tidak berlaku)

Tanpa mengetahui nama fitur, aplikasi metode-metode ini tetap memberikan wawasan berharga tentang fitur mana yang memiliki pengaruh lebih besar pada prediksi model. Memahami prinsip teknis di balik setiap metode memungkinkan pemilihan yang tepat berdasarkan karakteristik model dan kebutuhan analisis.

4. Desain Eksperimen untuk Pemilihan Hyperparameter Optimal

Pemilihan hyperparameter yang optimal sangat penting untuk memaksimalkan performa model. Berikut desain eksperimen komprehensif:

Strategi Pencarian Hyperparameter

1. Grid Search dengan Cross-Validation

Implementasi Teknis:

- Tentukan subset diskrit nilai untuk setiap hyperparameter (misalnya, learning_rate = [0.001, 0.01, 0.1, 1])
- Lakukan pencarian exhaustive di semua kombinasi hyperparameter

- Untuk setiap kombinasi, evaluasi model menggunakan k-fold cross-validation (biasanya k=5 atau k=10)
- Pilih kombinasi dengan performa rata-rata terbaik

Contoh untuk Decision Tree:

python
```
param_grid = {
   'max_depth': [3, 5, 7, 10, None],
   'min_samples_split': [2, 5, 10, 20],
   'min_samples_leaf': [1, 2, 4, 8],
   'criterion': ['squared_error', 'friedman_mse', 'absolute_error']
}
```

Tradeoff:

- **Komputasi**: Sangat intensif, kompleksitas meningkat secara eksponensial dengan jumlah hyperparameter
- **Stabilitas**: Sangat stabil dan deterministik
- **Generalisasi**: Dapat menghasilkan model yang over-optimized untuk validation set

2. Random Search dengan Cross-Validation

Implementasi Teknis:

- Tentukan distribusi (bukan nilai diskrit) untuk setiap hyperparameter
- Secara acak sample n kombinasi hyperparameter dari distribusi tersebut
- Evaluasi setiap kombinasi dengan k-fold cross-validation
- Pilih kombinasi dengan performa terbaik

Contoh untuk SGDRegressor:

python
```
param_distributions = {
   'alpha': loguniform(1e-5, 10),
   'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'],
   'eta0': loguniform(1e-5, 1),
   'penalty': ['l1', 'l2', 'elasticnet']
}
```

Tradeoff:

- **Komputasi**: Lebih efisien daripada Grid Search, terutama untuk dimensi tinggi
- **Stabilitas**: Kurang deterministik, tetapi biasanya memberikan hasil yang serupa antar running
- **Generalisasi**: Seringkali memberikan performa yang lebih baik daripada Grid Search dengan sumber daya komputasi yang sama

3. Bayesian Optimization

Implementasi Teknis:

- Mulai dengan beberapa evaluasi awal
- Bangun model probabilistik (biasanya Gaussian Process) yang memetakan hyperparameter ke performa
- Secara iteratif, optimasi acquisition function untuk menentukan titik berikutnya yang akan dievaluasi
- Perbarui model probabilistik setelah setiap evaluasi
- Hentikan setelah sejumlah iterasi atau ketika performa konvergen

Tradeoff:

- **Komputasi**: Lebih efisien daripada Grid/Random Search untuk hyperparameter yang mahal dievaluasi
- **Stabilitas**: Cenderung lebih stabil dibanding Random Search, meskipun masih ada elemen stokastik
- **Generalisasi**: Sangat baik untuk optimasi hyperparameter yang sulit, cenderung mengeksplorasi ruang secara lebih cerdas

Strategi Validasi untuk Generalisasi yang Baik

1. Nested Cross-Validation

Implementasi Teknis:

- Outer loop: k-fold CV untuk mengestimasi performa generalisasi
- Inner loop: k-fold CV untuk pemilihan hyperparameter
- Untuk setiap fold di outer loop, gunakan inner CV untuk memilih hyperparameter terbaik
- Latih model final dengan hyperparameter terbaik untuk fold tersebut
- Evaluasi pada outer fold test set

Kelebihan:

- Memberikan estimasi performa generalisasi yang tidak bias
- Mengurangi risiko data leakage
- Mendeteksi instabilitas pemilihan hyperparameter

Tradeoff:

- Sangat intensif komputasi (k^2 evaluasi model)
- Hyperparameter optimal dapat bervariasi antar outer fold

2. Time-Series Cross-Validation (untuk data deret waktu)

Implementasi Teknis:

- Gunakan expanding window atau sliding window CV
- Pastikan train set selalu mendahului test set secara temporal
- Untuk setiap split temporal, lakukan pemilihan hyperparameter

Kelebihan:

- Lebih realistis untuk data dengan ketergantungan temporal
- Menghindari look-ahead bias

Analisis Tradeoff Mendalam

1. Tradeoff Komputasi vs. Akurasi

Strategi Efisiensi:

- **Early stopping**: Hentikan evaluasi hyperparameter yang jelas inferior
- **Progressive refinement**: Mulai dengan pencarian kasar, lalu perhalus di area menjanjikan
- **Paralelisasi**: Manfaatkan paralelisme untuk mengevaluasi banyak kombinasi secara bersamaan
- **Transfer learning dari eksperimen sebelumnya**: Gunakan pengetahuan dari eksperimen serupa

Analisis Kuantitatif:

- Grid Search untuk n hyperparameter dengan m nilai per hyperparameter: O(m^n) evaluasi
- Random Search dengan B budget: O(B) evaluasi, biasanya B << m^n
- Bayesian Optimization: O(B log B) kompleksitas untuk akuisisi, plus evaluasi model

2. Tradeoff Stabilitas Pelatihan vs. Generalisasi

Untuk SGDRegressor:

- Learning rate tinggi: Konvergensi cepat tapi tidak stabil
- Learning rate rendah: Stabil tapi konvergensi lambat
- Early stopping: Dapat mencegah overfitting, tapi memerlukan validation set terpisah

Untuk Decision Tree:

- Max depth tinggi: Kapasitas model tinggi, risiko overfitting
- Min samples di leaf besar: Model lebih stabil, tapi mungkin underfitting

Strategi Pengujian Stabilitas:

- Jalankan pemilihan hyperparameter beberapa kali dengan seed berbeda
- Analisis varians performa antar running
- Perhatikan distribusi hyperparameter optimal yang ditemukan

3. Pertimbangan Praktis

Resource Constraints:

- Untuk dataset besar: Gunakan subset untuk pencarian awal, lalu validasi pada dataset penuh
- Untuk model yang lambat dilatih: Prioritaskan Random Search atau Bayesian Optimization
- Pertimbangkan pruning awal hyperparameter yang jelas tidak optimal

Domain Knowledge Integration:

- Tentukan range hyperparameter berdasarkan pengetahuan domain dan literatur
- Berikan bobot lebih pada area yang diketahui menjanjikan
- Gunakan validasi statistik untuk membedakan performa yang benar-benar berbeda dari noise

Eksperimen pemilihan hyperparameter yang well-designed harus menyeimbangkan kebutuhan eksplorasi space yang cukup luas sementara tetap komputasional feasible, sambil memastikan hasil yang diperoleh akan generalisasi dengan baik pada data baru.

5. Menangani Residual Plot Non-Linear dan Heteroskedastisitas dalam Linear Regression

Residual plot yang menunjukkan pola non-linear dan heteroskedastisitas mengindikasikan pelanggaran asumsi dasar linear regression, yang dapat menyebabkan estimasi parameter yang tidak efisien dan interval kepercayaan yang tidak valid. Berikut adalah langkah-langkah mendalam untuk mengatasi masalah ini:

1. Transformasi Variabel Target

Analisis Teknis: Transformasi variabel target (y) dapat mengatasi non-linearitas dan heteroskedastisitas secara simultan, terutama ketika varians residual meningkat seiring dengan nilai yang diprediksi.

Transformasi yang Umum:

- **Transformasi Logaritmik (log(y))**: Efektif ketika varians meningkat secara proporsional dengan nilai y. Transformasi ini menstabilkan varians dan "meluruskan" hubungan.
- **Transformasi Box-Cox**: $y(\lambda) = (y^\lambda - 1)/\lambda$ untuk $\lambda \neq 0$, dan $y(\lambda) = \log(y)$ untuk $\lambda = 0$. Parameter λ dapat dioptimalkan untuk memaksimalkan likelihood atau meminimalkan heteroskedastisitas.
- **Transformasi Akar Kuadrat ($\sqrt{y}$)**: Efektif untuk data count yang mengikuti distribusi Poisson, di mana varians sebanding dengan mean.
- **Transformasi Reciprocal (1/y)**: Berguna untuk kasus ekstrem di mana hubungan bersifat hiperbola.

Implementasi:

1. Uji beberapa transformasi (log, sqrt, inverse, Box-Cox dengan berbagai λ)
2. Untuk setiap transformasi, latih model dan plot residual
3. Pilih transformasi yang menghasilkan residual paling homoskedastik dan tidak berpola

Keterbatasan:

- Interpretabilitas menjadi lebih sulit
- Transformasi logaritmik memerlukan nilai $y > 0$
- Transformasi dapat mengubah skala error, sehingga perbandingan performa antar model menjadi kompleks

2. Transformasi Variabel Prediktor

Analisis Teknis: Pola non-linear dalam residual plot sering mengindikasikan bahwa hubungan antara prediktor dan target tidak linear, yang dapat diatasi dengan transformasi prediktor.

Strategi Transformasi:

- **Polynomial Features**: Menambahkan X^2, X^3, dll. untuk menangkap hubungan non-linear.
- **Spline Transformation**: Membagi range prediktor menjadi segmen dan memodelkan hubungan piece-wise linear atau polynomial, memberikan fleksibilitas lebih tinggi.
- **Transformasi Khusus Domain**: Berdasarkan pengetahuan domain, seperti transformasi logaritmik untuk variabel ekonomi, atau sinusoidal untuk data periodik.

Implementasi:

1. Analisis scatter plot antara setiap prediktor dan target
2. Identifikasi pola hubungan (kuadratik, kubik, logaritmik, eksponensial)
3. Terapkan transformasi yang sesuai
4. Evaluasi residual plot setelah transformasi

Kelebihan:

- Mempertahankan struktur linear regression, sehingga tetap interpretatif
- Dapat secara signifikan meningkatkan fit model tanpa beralih ke model yang lebih kompleks
- Lebih efisien komputasional dibandingkan model non-linear

Keterbatasan:

- Memerlukan eksplorasi data yang intensif
- Risiko overfitting jika terlalu banyak transformasi diterapkan
- Sulit untuk mengidentifikasi transformasi optimal untuk banyak prediktor

3. Weighted Least Squares (WLS) untuk Heteroskedastisitas

Analisis Teknis: WLS adalah metode estimasi yang memberikan bobot berbeda pada observasi berbeda berdasarkan varians residual. Observasi dengan varians residual lebih besar diberi bobot lebih kecil, dan sebaliknya.

Langkah-langkah Implementasi:

1. Fit model OLS (Ordinary Least Squares) awal
2. Hitung residual dan modelkan varians residual sebagai fungsi dari variabel prediktor atau nilai yang diprediksi
3. Tentukan bobot sebagai 1/varians untuk setiap observasi
4. Fit model WLS menggunakan bobot tersebut

Contoh Python:

python
```
# Fit model OLS awal
```

```
model_ols = sm.OLS(y, X).fit()
residuals = model_ols.resid

# Modelkan varians residual
abs_residuals = np.abs(residuals)
fitted_values = model_ols.fittedvalues
variance_model = sm.OLS(abs_residuals, sm.add_constant(fitted_values)).fit()

# Hitung bobot
weights = 1 / (variance_model.fittedvalues ** 2)

# Fit model WLS
model_wls = sm.WLS(y, X, weights=weights).fit()
```

Kelebihan:

- Estimator yang lebih efisien untuk koefisien
- Interval kepercayaan dan tes hipotesis yang lebih valid
- Mempertahankan interpretabilitas model linear

Keterbatasan:

- Memerlukan spesifikasi yang tepat dari struktur heteroskedastisitas
- Sensitif terhadap outlier
- Lebih kompleks secara komputasional dan interpretatif

4. Beralih ke Model yang Lebih Fleksibel

Analisis Teknis: Ketika transformasi dan WLS tidak adekuat, transisi ke model yang secara inheren non-linear atau robust terhadap heteroskedastisitas mungkin diperlukan.

Alternatif Model:

- **Generalized Additive Models (GAM)**: Memodelkan hubungan non-linear dengan fungsi smoothing untuk setiap prediktor, memberikan fleksibilitas yang tinggi namun tetap interpretatif.
- **Gradient Boosting Regression**: Sangat kuat dalam menangkap hubungan non-linear dan interaksi kompleks tanpa spesifikasi eksplisit.
- **Support Vector Regression dengan Kernel Non-linear**: Memetakan data ke dimensi yang lebih tinggi di mana hubungan menjadi linear.
- **Quantile Regression**: Memodelkan berbagai kuantil dari distribusi bersyarat target, ideal untuk heteroskedastisitas.

Pertimbangan Implementasi:

1. Mulai dari model yang lebih sederhana dan tingkatkan kompleksitas secara gradual
2. Gunakan validasi silang untuk mencegah overfitting
3. Regularisasi diperlukan untuk model yang lebih kompleks

4. Pertimbangkan tradeoff antara performa dan interpretabilitas

Kelebihan:

- Dapat menangkap hubungan yang sangat kompleks
- Sering memberikan performa prediktif yang superior
- Beberapa model (seperti GAM) tetap menawarkan interpretabilitas parsial

Keterbatasan:

- Risiko overfitting lebih tinggi
- Lebih sulit diinterpretasi (terutama ensemble dan deep learning)
- Memerlukan lebih banyak data untuk estimasi yang stabil
- Komputasi lebih intensif

5. Pendekatan Hybrid dan Diagnostik Lanjutan

Strategi Hybrid:

- **Segmented Regression**: Membagi data menjadi region berbeda dan menerapkan model linear yang berbeda untuk setiap region.
- **Transform-Both-Sides (TBS)**: Menerapkan transformasi yang sama pada kedua sisi persamaan regresi.
- **Kombinasi transformasi dan WLS**: Menerapkan transformasi untuk mengatasi non-linearitas, lalu WLS untuk residual heteroskedastisitas yang tersisa.

Diagnostik Lanjutan:

- **Test Breusch-Pagan/White**: Untuk mendeteksi heteroskedastisitas
- **Test Ramsey RESET**: Untuk mendeteksi kesalahan spesifikasi model/non-linearitas
- **Partial Residual Plots**: Untuk mengidentifikasi hubungan non-linear untuk prediktor individual
- **Q-Q Plot Residual**: Untuk memeriksa normalitas residual

Strategi Validasi:

1. Validasi silang dengan prediksi interval untuk menilai kalibrasi model
2. Pengujian bootstrap untuk estimasi interval kepercayaan yang robust
3. Analisis sensitivitas untuk menguji robustness terhadap outlier dan asumsi model

Pilihan pendekatan optimal bergantung pada severitas non-linearitas dan heteroskedastisitas, ukuran dataset, kebutuhan interpretabilitas, dan domain aplikasi. Pendekatan yang paling menyeluruh biasanya adalah memulai dengan perbaikan model linear (transformasi dan WLS), lalu beralih ke model yang lebih fleksibel jika diperlukan, dengan validasi ketat di setiap langkah.