PREDIKSI VALIDITAS ALAMAT BERBASIS *MACHINE LEARNING PADA WEB FORMULIR* DI PT TRIMEGAH ASSET MANAGEMENT

[1]Evy Nurmiati, [2] Muhammad Zikri Pasa
[1,2]Sistem Informasi, Fakultas Sains dan Teknologi, UIN Syarif Hidayatullah Jakarta,
Jl. Ir. H. Djuanda No. 95, Ciputat, Tangerang Selatan, Banten, Indonesia 15412
Email: evy.nurmiati@uinjkt.ac.id, muhammad.zikripasa22@mhs.uinjkt.ac.id

ABSTRAK

Kebutuhan akan integritas data merupakan pilar fundamental dalam menjaga stabilitas sistem keuangan nasional. Namun, proses pendataan nasabah di banyak perusahaan yang masih mengandalkan entri data manual sangat rentan terhadap kesalahan manusia (*human error*), khususnya di PT. Trimegah Asset Management. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan dan membangun sebuah web formulir pendataan nasabah berbasis *machine learning* untuk memvalidasi data nasabah, khususnya data alamat. Dataset yang digunakan terdiri dari 50004 baris alamat dari *database* internal perusahaan. Metode yang digunakan adalah pengembangan model *machine learning* menggunakan algoritma random forest dan integrasi antara model *machine learning* dengan menggunakan *framework* Flask API. Hasil penelitian menunjukkan bahwa model *machine learning* yang dibangun memiliki akurasi data sebesar 99%, mampu terintegrasi dengan sistem yang dibangun. Sistem tersebut mampu mengotomatisasi verifikasi data serta meningkatkan validitas *entry data*.

***Keywords*:** Flask API, *Machine Learning*, Validasi Alamat, Web Formulir Nasabah

1 PENDAHULUAN

Pertumbuhan pesat sektor jasa keuangan di Indonesia, didorong oleh digitalisasi dan peningkatan akses investasi, telah menempatkan peran manajer investasi dalam posisi yang krusial sebagai pengelola dana nasabah.[1] Kebutuhan akan integritas data yang tinggi di industri keuangan tidak lagi sekadar masalah efisiensi internal, melainkan merupakan mandat kepatuhan yang mengikat secara hukum.[2] Meskipun terdapat tuntutan regulasi yang ekstrem terhadap kualitas data, proses pendataan nasabah di banyak perusahaan, terutama yang masih mengandalkan entri data manual atau semi-otomatis, tetap rentan terhadap kesalahan manusia (*human error*).[3] Permasalahan tersebut dihadapi oleh PT Trimegah Asset Management terkait pendataan nasabah dimana banyak nasabah yang memasukkan alamat rumah atau alamat kantor secara sembarangan sehingga data tidak lengkap atau salah format. Oleh karena itu dirancanglah sebuah web formulir pendataan nasabah yang mampu memvalidasi alamat yang diinput menggunakan fitur *machine learning* secara otomatis. Implementasi teknologi *machine learning* dan *Artificial Intelligence* (AI) merupakan langkah strategis yang diperlukan untuk menjembatani kesenjangan antara tuntutan akurasi regulasi yang sangat tinggi dan keterbatasan proses input data secara manual. Penerapan *web* berbasis *machine learning* juga memberikan peluang besar untuk *enhanced scalability* (peningkatan skalabilitas).[4] Dengan demikian, perancangan website adalah langkah penting dalam melakukan modernisasi proses bisnis, mengurangi inkonsistensi data, dan memastikan bahwa perusahaan dapat mengelola pertumbuhan sambil mempertahankan tingkat kepatuhan data tertinggi.[5]

Tujuan penelitian secara umum adalah :Merancang dan membangun sebuah aplikasi web formulir pendataan nasabah yang dapat mengurangi kesalahan input data serta mengotomatisasi verifikasi validitas alamat nasabah. Mengembangkan dan mengimplementasikan model *machine learning* berbasis algoritma *classification* untuk memvalidasi dan meningkatkan akurasi data nasabah secara *real-time*. Mengintegrasikan model *machine learning* ke dalam sistem web formulir menggunakan *Application Programming Interface* (API) untuk memastikan validasi data yang efisien.

2 TINJAUAN PUSTAKA

Machine Learning (ML) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data dan meningkatkan kinerjanya seiring waktu tanpa diprogram secara eksplisit untuk setiap aturan. Definisi formal yang banyak dirujuk dalam literatur akademik dikemukakan oleh Mitchell (1997), yang menyatakan bahwa sebuah program komputer dikatakan belajar dari pengalaman *E* sehubungan dengan kelas tugas *T* dan ukuran kinerja *P*, jika kinerjanya pada tugas di *T*, sebagaimana diukur oleh *P*, meningkat dengan pengalaman *E*.[6] Dalam taksonomi pembelajaran mesin, pendekatan yang paling umum diterapkan dalam industri adalah *Supervised Learning*. Metode ini melibatkan pelatihan model menggunakan dataset yang memiliki label (pasangan input-output), di mana tujuannya adalah memetakan fungsi input ke output untuk memprediksi label pada data baru yang belum terlihat.

Pengembangan sistem ML yang efektif mengikuti metodologi terstruktur untuk menjamin kualitas dan reproduktibilitas. Rao (2019) dan Hulten (2018) menguraikan tahapan kritis dalam siklus hidup ini, yang meliputi definisi masalah, persiapan data, pengembangan model, evaluasi, hingga penyebaran (deployment).[7] Data mentah di dunia nyata sering kali tidak lengkap atau mengandung *noise*. Tahap *data cleaning* melibatkan penanganan nilai yang hilang (*missing values*) melalui metode imputasi (mengisi nilai rata-rata/median) atau penghapusan, serta identifikasi *outliers* yang dapat mendistorsi model.[8] Setelah dibersihkan, data harus melalui tahap *preprocessing* dan *feature engineering*. Proses ini mencakup normalisasi skala fitur numerik dan pengkodean variabel kategorikal (*encoding*) agar dapat diproses secara matematis oleh algoritma.[9] Praktisi data umumnya menghabiskan sebagian besar waktu proyek pada fase ini karena kualitas data berkorelasi langsung dengan kinerja model.[8] *Random Forest* adalah algoritma *ensemble learning* yang diperkenalkan oleh Leo Breiman (2001). Algoritma ini membangun sekumpulan pohon keputusan (*decision trees*) selama pelatihan dan menghasilkan output berupa kelas yang merupakan modus (klasifikasi) atau rata-rata (regresi) dari pohon-pohon individu.[10]

Keunggulan utama *Random Forest* terletak pada penerapan metode *bagging* (*bootstrap aggregating*) dan pemilihan fitur acak. Dengan melatih setiap pohon pada subset data yang berbeda dan hanya mempertimbangkan subset fitur acak pada setiap pemecahan node, *Random Forest* secara efektif mengurangi varians dan risiko *overfitting* yang sering terjadi pada pohon keputusan tunggal.[11] Hal ini menjadikannya algoritma yang *robust* untuk berbagai jenis dataset, termasuk yang memiliki dimensi tinggi.[12] Evaluasi model klasifikasi didasarkan pada *Confusion Matrix*, yang memetakan prediksi model terhadap label aktual menjadi *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).[13]

1. Accuracy (Akurasi): Mengukur rasio prediksi benar terhadap total data

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

   Akurasi dapat menjadi bias pada dataset yang tidak seimbang (imbalanced dataset).[13]
2. Precision (Presisi): Mengukur ketepatan prediksi positif. Relevan ketika biaya kesalahan positif palsu (FP) tinggi

$$Precision = \frac{TP}{TP + FP}$$

3. Recall (Sensitivitas): Mengukur kemampuan model menemukan semua kasus positif. Relevan ketika biaya kesalahan negatif palsu (FN) tinggi, seperti pada diagnosis medis.[13]

$$Recall = \frac{TP}{TP + FN}$$

4. F1 Score: Merupakan rata-rata harmonik dari presisi dan recall. Metrik ini memberikan evaluasi yang lebih seimbang daripada akurasi, terutama ketika distribusi kelas tidak merata.[14]

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Salah satu tantangan utama dalam pemodelan adalah *overfitting*, yaitu kondisi di mana model mempelajari detail dan *noise* pada data latih sedemikian rupa hingga gagal melakukan generalisasi pada data baru.[15] Gejalanya adalah performa tinggi pada data latih namun rendah pada data uji. Untuk memitigasi risiko ini dan mendapatkan estimasi kinerja yang lebih valid, digunakan teknik *K-Fold Cross-Validation.* Dalam metode ini, dataset dibagi menjadi bagian K (*folds*). Proses pelatihan dan validasi diulang sebanyak K kali, di mana setiap *fold* digunakan satu kali sebagai data validasi sementara sisanya sebagai data latih.[8] Teknik ini memastikan bahwa setiap data point digunakan untuk pengujian, sehingga mengurangi bias yang mungkin muncul dari pembagian data statis (*hold-out method*).[16] Tahap akhir dalam siklus hidup ML adalah *deployment*, di mana model diintegrasikan ke dalam lingkungan produksi untuk digunakan oleh pengguna akhir. Disiplin ini sering disebut sebagai MLOps (*Machine Learning Operations*), yang bertujuan menyatukan pengembangan sistem ML dengan operasi sistem.[10]

Integrasi model ML dengan aplikasi web atau *mobile* umumnya menggunakan arsitektur *Client-Server* melalui antarmuka pemrograman aplikasi (API) berbasis REST (*Representational State Transfer*). Dalam arsitektur ini, model ML ditempatkan di sisi *server* (backend) dan diekspos melalui titik akhir (*endpoints*) URL. Aplikasi *client* (frontend) mengirimkan data input melalui permintaan HTTP (biasanya metode POST), dan server merespons dengan hasil prediksi dalam format standar seperti JSON. Salah satu alat yang populer untuk implementasi layanan mikro (*microservices*) ML berbasis Python adalah Flask. Grinberg (2014) mendeskripsikan Flask sebagai kerangka kerja mikro yang ringan namun fleksibel, memungkinkan pengembang untuk membangun layanan web dengan cepat tanpa struktur yang kaku. Flask memfasilitasi pembuatan *routes* untuk menangani permintaan prediksi, memuat model yang telah dilatih (misalnya format .pkl atau .joblib), dan mengembalikan hasil inferensi ke sistem eksternal secara efisien.[17]

3 METODE PENELITIAN

3.1 Dataset

Dataset yang digunakan terdiri dari 50.004 baris data alamat nasabah yang diperoleh dari database internal PT Trimegah Asset Management yang dikonversi dalam format csv. Data alamat mencakup alamat rumah dan alamat kantor dengan variasi format penulisan.

3.2 Alur Pembangunan Model

1. Proses Data Cleaning: Dataset yang diimpor database akan dibersihkan menggunakan sintaks untuk memeriksa *missing value*, mengatasi *missing value* dengan *dropping* atau *imputation*, memeriksa *duplicated data*, dan membersihkan duplikat.
2. Proses Pelabelan: Proses pelabelan dilakukan menggunakan *script python* yang mengidentifikasi apakah suatu alamat mengandung angka (nomor jalan dan kode pos) atau entitas alamat seperti : "jalan', "jl", "jl." "rt", "kec", dan lain-lain. *Script* akan mengelompokkan validitas alamat berdasarkan standar penulisan alamat yang lengkap, yaitu mencakup nama jalan, nomor rumah, RT/RW, kelurahan, kecamatan, kab/kota, provinsi, dan kode pos. Alamat yang memenuhi format lengkap akan diberi label "Valid", sedangkan alamat yang tidak lengkap, ambigu, atau tidak sesuai format akan diberi label "Kurang Valid".
3. *Preprocessing Data*: Proses pre-processing data dilakukan untuk mengubah data mentah menjadi format yang sesuai dan kompatibel agar dapat dipahami dan diolah oleh algoritma nantinya. Dalam hal ini, akan dilakukan label encoding dan vektorisasi. Label encoding pada data akan mengganti label "Valid" menjadi angka 1 dan "Kurang Valid" menjadi angka 0.
4. Pada tahap data splitting, data dibagi menjadi *data training, data testing*, dan *data validation* dengan rasio 80:10:10. Rasio ini menghasilkan distribusi data yang proporsional untuk pelatihan, pengujian, dan validasi model.

5. Proses vektorisasi dilakukan menggunakan metode TF-IDF *(Term Frequency–Inverse Document Frequency)* untuk mengubah data alamat berbasis teks menjadi representasi numerik. Metode ini memberikan bobot pada setiap kata berdasarkan frekuensi kemunculannya dalam satu dokumen dan tingkat keunikannya terhadap keseluruhan dataset. Dengan demikian, kata-kata yang bersifat umum akan memiliki bobot lebih rendah, sedangkan kata-kata yang lebih spesifik dan informatif akan memiliki bobot lebih tinggi.
6. *Model Development* dan Evaluasi
   Model klasifikasi dibangun menggunakan algoritma *Random Forest Classifier* karena stabil terhadap *noise* dan mampu menangani fitur berdimensi tinggi. Evaluasi dilakukan menggunakan metrik akurasi, *precision, recall, F1-score,* serta *confusion matrix*.

3.3 Web Application

Web application dibangun dengan mengintegrasikan model *machine learning* dengan *front-end* yang berbasis HTML, CSS, dan JavaScript menggunakan Flask sebagai REST API di sisi *backend*. Model *machine learning* yang dibangun disimpan dalam bentuk *file* .pkl menggunakan model serialisasi (*pickle*) agar dapat dimuat kembali dan digunakan pada aplikasi. Struktur folder aplikasi terdiri dari *file* model.pkl, api.py, script.js, style.css, dan index.html.

4 HASIL DAN PEMBAHASAN

4.1 Hasil Pelabelan

Proses pelabelan menghasilkan dua kelas, "Valid" dan "Kurang Valid" yang dikonversi ke dalam angka 1 dan 0. Hasil ini menunjukkan bahwa proses pelabelan dengan *script python* mampu membedakan kualitas alamat secara sistematis seperti yang ditunjukan tabel 2 berikut.

Tabel 1 Hasil Pelabelan

No.	Alamat	Result
1	Gedung Graha Irama Lt 15, Jalan HR. Rasuna Said Blok X-1 Kav 1-2, RT 006/004, Kuningan Timur, Setiabudi, Jakarta Selatan	Valid
2	Jl Kko Raya Cilandak Pasar Minggu	Kurang Valid

4.2 Hasil Preprocessing Data

Label encoding mengubah data kategorikal pada data menjadi numerik. Rincian contoh hasil encoding ditampilkan pada tabel 3, sedangkan contoh hasil transformasi fitur menggunakan TF-IDF ditunjukkan pada tabel 4 yang memperlihatkan contoh vektor hasil ekstraksi fitur.

Tabel 2 Hasil Label Encoding

No.	Alamat	Hasil
1	Gedung HKBP Lt II, Jl. Uskup Agung Sugio Pranoto No. 6, Kel. Madras Hulu, Kec. Medan Polonia	1
2	Desa Sanggrahan Grogol Sukoharjo	0

Tabel 3 Hasil Vektorisasi

Coords	*Values*
(0, 20520)	0.18506154806786426
...	⋮
(4002, 8241)	0.4571207539915973

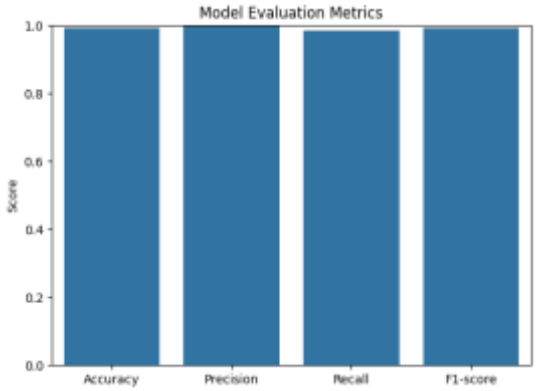
4.3 Hasil Evaluasi Model

Model Random Forest Classifier menunjukkan performa yang baik dalam mengklasifikasikan alamat ke dalam kelas Valid dan Kurang Valid. Gambar 2 merupakan grafik perbandingan metrik evaluasi . Evaluasi kinerja model dilakukan menggunakan beberapa metrik utama, yaitu: *accuracy, precision, recall, F1-score,* serta *confusion matrix.* Grafik menunjukkan bahwa seluruh metrik berada

pada rentang nilai yang sangat tinggi dan relatif seimbang, yang menegaskan kualitas performa model secara menyeluruh.

Tabel 4 Hasil Evaluasi Metrik

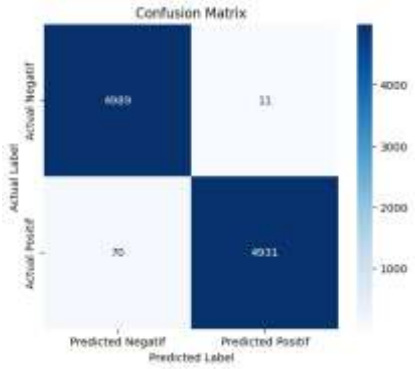
Metrik	Nilai
Accuracy	0.9919 (99,19%)
Precision	0.9978 (99,78%)
Recall	0.9860 (98,60%)
F1-Score	0.9919 (99,19%)



Gambar 2 Visualisasi Hasil Evaluasi Model

Berdasarkan hasil yang ditunjukkan pada Tabel 5, model memperoleh nilai akurasi sebesar 99,19%, yang menunjukkan bahwa hampir seluruh data berhasil diklasifikasikan secara benar. Nilai akurasi yang sangat tinggi ini mengindikasikan bahwa model mampu mempelajari pola karakteristik alamat secara efektif dan konsisten, dengan tingkat kesalahan yang sangat rendah, yaitu sekitar 0,81%. Selanjutnya, nilai *precision* sebesar 99,78% menunjukkan bahwa hampir seluruh data yang diprediksi sebagai alamat valid oleh model memang benar-benar merupakan alamat valid. Tingginya precision ini menandakan bahwa risiko kesalahan dalam memvalidasi alamat yang sebenarnya tidak valid (*false positive*) sangat kecil, sebagaimana tercermin pada nilai *false positive* yang rendah pada Gambar 3. Hal ini sangat penting dalam konteks sistem validasi alamat, karena kesalahan jenis ini dapat berdampak langsung terhadap kualitas data dan proses bisnis lanjutan.

Sementara itu, nilai *recall* sebesar 98,60% mengindikasikan bahwa model mampu mengidentifikasi hampir seluruh alamat valid yang terdapat dalam data uji. Dengan demikian, hanya sebagian kecil alamat valid yang gagal dikenali oleh model (*false negative*), sebagaimana ditampilkan pada Gambar 3. Tingginya *recall* ini mencerminkan sensitivitas model yang baik terhadap kelas positif, sehingga potensi kehilangan data alamat valid dapat diminimalkan. Nilai *F1-score* sebesar 99,19%, yang juga ditampilkan pada Tabel 5, menunjukkan bahwa model memiliki keseimbangan yang sangat baik antara *precision* dan *recall*. Hal ini mengindikasikan bahwa model tidak hanya unggul dalam meminimalkan kesalahan prediksi positif, tetapi juga efektif dalam menangkap mayoritas data positif yang sebenarnya, sehingga performa model dapat dikategorikan stabil dan andal secara keseluruhan.



Gambar 3 Confusion Matrix

Analisis lebih lanjut dilakukan menggunakan visualisasi *confusion matrix* yang ditampilkan pada Gambar 3, yang menunjukkan bahwa terdapat 4931 data *true positive* dan 4989 data *true negative*, yang masing-masing merupakan prediksi benar untuk kelas alamat valid dan tidak valid. Jumlah kesalahan relatif sangat kecil, dengan hanya 70 data *false negative* dan 11 data *false positive*. Distribusi ini memperlihatkan bahwa model cenderung lebih berhati-hati dalam memprediksi kelas positif, yang tercermin dari jumlah *false positive* yang sangat rendah. Karakteristik ini sangat menguntungkan bagi sistem validasi alamat, karena lebih aman menolak sebagian kecil alamat valid dibandingkan menerima alamat yang tidak valid.

4.4 Evaluasi Cross Validation

Terdapat dugaan model yang dibangun mengalami *overfitting*. Untuk itu, dilakukan evaluasi menggunakan metode *cross-vali*dation lima lipatan (*5-fold cross validation*).

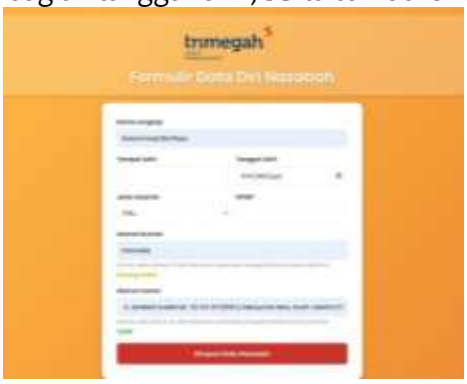
Table 5 Hasil Cross Validation

Metrik	Nilai
Fold 1	0.99390061
Fold 2	0.99290071
Fold 3	0.99390061
Fold 4	0.99330067
Fold 5	0.9915
Rata-rata	0.9931005199480053

Hasil pengujian menunjukkan nilai akurasi yang konsisten pada setiap *fold*, dengan rata-rata sebesar 99,31%, yang sejalan dengan nilai akurasi pada data uji sebagaimana ditampilkan pada Tabel 6. Terlihat bahwa nilai setiap *fold* stabil dan tidak ada penurunan ekstrim. Konsistensi ini menegaskan bahwa model tidak mengalami *overfitting* dan memiliki kemampuan generalisasi yang sangat baik terhadap data yang belum pernah dilihat sebelumnya.

4.5 Hasil Implementasi *Web Application*

Model yang telah dikembangkan berhasil diintegrasikan ke dalam *web application* berbasis Flask sebagai REST API. Model yang disimpan dalam *file* .pkl yang dapat dimuat kembali dan digunakan untuk melakukan prediksi secara real-time melalui *front-end* berbasis web. Gambar 4 yang menampilkan *front-end* halaman depan web aplikasi. Terlihat beberapa kolom teks seperti : nama, tempat lahir, NPWP, alamat rumah, dan alamat kantor. Terdapat *radio buttons* untuk bagian kelamin dan *date picker* untuk bagian tanggal lahir, serta tombol untuk submit data.



Gambar 4. Tampilan Antarmuka

Gambar 5. Hasil Prediksi

Pada gambar 5 menunjukkan bahwa sistem dapat menerima input alamat dan memprediksi validitas alamat dengan baik. Aplikasi web menampilkan hasil prediksi secara *real-time* baik "Valid" maupun "Kurang Valid". Web aplikasi yang dibangun memiliki *user interface* yang linear dengan logo perusahaan serta mampu memprediksi validitas alamat dengan cepat dan tepat.

5 KESIMPULAN

Berdasarkan hasil penelitian dan pengujian yang telah dilakukan, dapat disimpulkan bahwa model *machine learning* berbasis *Random Forest Classifier* mampu melakukan klasifikasi validitas alamat secara sangat efektif dan andal. Proses pelabelan dan *preprocessing data*, yang meliputi *label encoding* serta ekstraksi fitur menggunakan TF-IDF, berhasil mengubah data teks alamat menjadi representasi numerik yang dapat diproses dengan baik oleh model. Hasil evaluasi menunjukkan performa yang sangat tinggi dengan nilai *accuracy* sebesar 99,19%, *precision* 99,78%, *recall* 98,60%, dan *F1-score* 99,19%, yang menandakan keseimbangan optimal antara kemampuan mengenali alamat valid dan meminimalkan kesalahan klasifikasi. Selain itu, hasil *cross-validation* dengan rata-rata akurasi sebesar 99,31% memperkuat bahwa model memiliki generalisasi yang baik dan tidak mengalami *overfitting*. Keberhasilan integrasi model ke dalam *web application* berbasis Flask sebagai REST API juga membuktikan bahwa sistem ini tidak hanya unggul secara teoritis, tetapi juga siap diimplementasikan secara praktis dalam lingkungan nyata.

Adapun saran untuk pengembangan selanjutnya, penelitian ini dapat ditingkatkan dengan memperluas variasi dan jumlah dataset alamat agar model semakin *robust* terhadap berbagai pola dan format alamat yang lebih kompleks. Selain itu, eksplorasi metode *feature engineering* lain atau penggunaan model berbasis *deep learning* seperti LSTM atau *transformer* dapat dipertimbangkan untuk membandingkan performa dengan pendekatan *Random Forest*. Dari sisi sistem, pengembangan antarmuka pengguna dapat diperluas dengan fitur validasi tambahan, seperti *auto-correction* atau rekomendasi perbaikan alamat, serta integrasi dengan basis data geografis atau layanan pemetaan untuk meningkatkan akurasi validasi secara spasial. Dengan demikian, sistem validasi alamat yang dibangun diharapkan dapat semakin optimal dan aplikatif untuk mendukung kebutuhan industri maupun institusi berbasis data.

REFERENSI

[1] O. Alifa, Putri Riska; Ferli, "Pengaruh Kemampuan Manajer Investasi terhadap Kinerja Reksa Dana Pendapatan Tetap di Indonesia," 2022.

[2] A. Fariah, "Mengeksplorasi Masa Depan Audit: Memanfaatkan Teknologi dan Analisis Data untuk Peningkatan Integritas Keuangan," *Cakrawala*, vol. 6, 2023.

[3] N. Hanifah, M. Irwan, and P. Nasution, "Manajemen Data Yang Efektif : Solusi Untuk Mencegah dan Mengatasi Duplikasi Data Dalam Perusahaan," vol. 3, no. 1, 2025.

[4] N. I. Zaki and A. Qoiriah, "Implementasi Algoritma Machine Learning untuk Pengelolaan Laporan pada Website Dinas Penanaman Modal Dan Pelayanan Terpadu Satu Pintu (DPMPTSP) Kota Surabaya," vol. 06, pp. 1063–1075, 2025.

[5] R. Anggara, "Digitalisasi Manajemen Sistem Informasi Pengarsipan Perpustakaan," vol. 1, no. 2, pp. 86–92, 2025.

[6] T. M. Jordan, M. I., & Mitchell, *Machine Learning: Algorithms, Real-World Applications and Research Directions*. 2015. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7983091/

[7] L. Breiman, "Random Forests." [Online]. Available: https://www.scirp.org/reference/referencespapers?referenceid=1734556

[8] L. O. B. Vlad Teodorescu, "Assessing the Validity of k-Fold Cross-Validation for Model Selection: Evidence from Bankruptcy Prediction Using Random Forest and XGBoost,"

Multidiscip. Digit. Publ. Inst., 2025, [Online]. Available: https://www.mdpi.com/2079-3197/13/5/127

[9] C. G. W. Drew Wilimitis, "Practical Considerations and Applied Examples of Cross-Validation for Model Development and Evaluation in Health Care: Tutorial," *J. Biomed. Informatics (NIH / PMC)*, 2024, [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11041453/

[10] S. H. Dominik Kreuzberger, Niklas Kühl, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *arXiv*, 2022, [Online]. Available: https://arxiv.org/abs/2205.02302

[11] T. Mitchell., "Machine learning," 2026, [Online]. Available: https://www.bibsonomy.org/bibtex/479a66c32badb3a455fbdcf8e6633a5d

[12] A. W. T. Haokun Dong, Rui Liu, "Accuracy Comparison between Five Machine Learning Algorithms for Financial Risk Evaluation," *J. Risk Financ. Manag.*, 2024, [Online]. Available: https://www.mdpi.com/1911-8074/17/2/50

[13] Google Developers, "Classification: Accuracy, Recall, Precision, and Related Metrics," Google Machine Learning Crash Course.

[14] S. Developers, "f1_score — scikit-learn Documentation," scikit-learn Documentation.

[15] Google Developers, "Overfitting," Google Machine Learning Crash Course. [Online]. Available: https://developers.google.com/machine-learning/crash-course/overfitting/overfitting

[16] "K-Fold Cross Validation in Machine Learning," The Knowledge Academy. [Online]. Available: https://www.theknowledgeacademy.com/blog/k-fold-cross-validation-in-machine-learning/

[17] "Deploy Machine Learning Model using Flask," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/deploy-machine-learning-model-using-flask/