Pengaruh Penggunaan Platform CodeEasy terhadap Tingkat Pemahaman Mahasiswa dalam Pembelajaran Data Science Python

Evaluating the Use of the CodeEasy Platform on Students' Understanding of Data Science Python

Galur Arasy Lumintang*[1], Usman Nurhasan[2], Vivin Ayu Lestari[3]
Program Studi Sistem Informasi Bisnis, Jurusan Teknologi Informasi, Politeknik Negeri Malang, +62341404424
E-mail : adamhenderson3x3@gmail.com[1], usmannurhasan@polinema.ac.id[2], vivin.ayu@polinema.ac.id[3]*
**Corresponding author*
Received 8 September 2025; Revised 21 October 2025; Accepted 4 November 2025

Abstrak-Penelitian ini bertujuan menguji hipotesis bahwa penggunaan platform pembelajaran mandiri CodeEasy dapat meningkatkan pemahaman mahasiswa pada mata kuliah Data Science Python. CodeEasy merupakan lingkungan belajar asinkron yang menyediakan latihan berbasis kode dengan penilaian otomatis melalui test case, serta umpan balik instan untuk mendorong keterlibatan aktif mahasiswa. Penelitian menggunakan desain one-group pretest–posttest dengan 28 mahasiswa Program Studi Sistem Informasi Bisnis, Politeknik Negeri Malang, semester genap 2024/2025. Analisis Paired Sample t-test menunjukkan peningkatan signifikan skor pemahaman mahasiswa (rata-rata pretest = 32,34%; posttest = 80,38%; $t(27) = 9{,}078, p < 0{,}001$), dengan ukuran efek besar (Cohen's $d = 1{,}72$), yang mendukung hipotesis penelitian. Hasil ini bersifat indikatif, mengingat desain tanpa kelompok kontrol membatasi kesimpulan kausalitas dan generalisasi. Penelitian ini memberikan bukti awal mengenai efektivitas platform autograding dalam mendukung pembelajaran mandiri pemrograman, dan menyarankan penerapan lebih lanjut dengan desain kontrol serta integrasi Explainable AI (XAI) untuk meningkatkan transparansi evaluasi kode dan personalisasi umpan balik.

Kata Kunci - Autograding, CodeEasy, Data Science Python, Evaluasi Kode, Pembelajaran Mandiri

Abstract-*This study aims to test the hypothesis that the use of the self-paced learning platform CodeEasy can improve students' understanding in the Data Science Python course. CodeEasy is an asynchronous learning environment that provides code-based exercises with automated assessment through test cases, along with instant feedback to promote active student engagement. The study employed a one-group pretest–posttest design involving 28 students from the Business Information Systems Study Program at Politeknik Negeri Malang during the even semester of 2024/2025. Paired Sample t-test analysis showed a significant increase in students' understanding scores (pretest mean = 32.34%; posttest mean = 80.38%; t(27) = 9.078, p < 0.001), with a large effect size (Cohen's d = 1.72), supporting the research hypothesis. These results are indicative, as the lack of a control group limits causal conclusions and generalization. This study provides preliminary evidence of the effectiveness of an autograding platform in supporting self-paced programming learning and suggests further applications using a controlled design.*

Keywords*: Autograding, Data Science Python, Code Evaluation, CodeEasy, Self-Paced Learning*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

1. PENDAHULUAN

Transformasi digital telah membawa perubahan mendasar dalam paradigma pendidikan tinggi, khususnya dalam pembelajaran pemrograman dan data science. Pandemi COVID-19 semakin mempercepat adopsi teknologi pembelajaran daring secara luas, mendorong institusi pendidikan untuk mengembangkan platform yang adaptif, efektif, dan mampu menghasilkan high-impact contribution terhadap proses belajar mahasiswa[1], [2]. Dalam konteks pembelajaran Python for Data Science, mahasiswa sering menghadapi kesulitan dalam memahami konsep abstrak serta logika komputasional yang kompleks, yang menjadi hambatan signifikan terhadap keberhasilan pembelajaran daring[3], [4]. Berbagai penelitian menunjukkan bahwa efektivitas pembelajaran daring sangat dipengaruhi oleh rancangan platform dan metodologi pengajarannya, sehingga dibutuhkan inovasi sistem yang tidak hanya mendukung proses belajar, tetapi juga mampu menyediakan mekanisme asesmen otomatis yang andal, valid, dan novel high-impact bagi pengembangan pendidikan tinggi.

Perkembangan teknologi automated assessment telah membuka peluang baru dalam evaluasi pembelajaran pemrograman dengan tingkat akurasi dan efisiensi yang tinggi. Sejumlah penelitian menunjukkan kemajuan signifikan dalam pengembangan teknik dan sistem asesmen otomatis [5], [6],[7], [8], [9]. Platform pemrograman daring kini menjadi tren sebagai media interaktif yang mengintegrasikan umpan balik otomatis untuk meningkatkan hasil belajar dan memberikan kontribusi novelty high-impact dalam praktik pengajaran [10], [11], [12]. Messer et al., melalui systematic review terhadap 121 publikasi, menemukan bahwa 65,07 % studi melaporkan automated feedback meningkatkan performa siswa, dengan korelasi kuat antara penilaian otomatis dan manual ($r = 0{,}789$; $p < 0{,}001$) [6]. Studi longitudinal oleh Skalka dan Drlik menunjukkan korelasi positif antara aktivitas automated assessment dan performa ujian menggunakan Virtual Programming Lab, dengan rentang 0,512–0,721[13]. Penelitian Orvalho et al. pada 528 mahasiswa melalui platform GitSEED juga melaporkan bahwa 91,8 % responden menilai asesmen otomatis bermanfaat dalam pembelajaran pemrograman [14]. Temuan-temuan ini menegaskan potensi asesmen otomatis sebagai high-impact contribution yang mampu meningkatkan efisiensi evaluasi sekaligus hasil belajar mahasiswa secara signifikan.

Meskipun demikian, sejumlah penelitian terkini mengidentifikasi adanya kesenjangan penting dalam implementasi automated assessment. Figueras et al. menyoroti bahwa sebagian besar studi masih terbatas pada satu sistem tanpa perbandingan lintas konteks, yang menunjukkan gap penelitian signifikan terkait generalisasi hasil. Messer et al. menemukan bahwa rule-based automated assessment tools belum mampu menilai tugas pemrograman terbuka yang bersifat kompleks, sedangkan Vinueza-Morales et al. melalui analisis bibliometrik menegaskan dominasi penelitian dari negara maju dengan minimnya studi di negara berkembang dan konteks pendidikan alternatif, yang menegaskan perlunya inovasi high-impact dalam konteks lokal. Alghamdi menunjukkan bahwa mayoritas studi e-learning masih menggunakan analisis deskriptif tanpa uji statistik inferensial yang memadai [15]. Sauerwein et al. menambahkan bahwa pemahaman terhadap faktor kesuksesan Automated Programming Assessment Systems (APASs) masih terbatas akibat minimnya replikasi penelitian lintas sistem [16]. Bednarowska-Michaiel et al.menyoroti tantangan pengukuran efektivitas pembelajaran data science interdisipliner yang menggabungkan keterampilan statistik dan komputasional [17], sementara Rump et al. menegaskan perlunya penelitian lanjutan untuk asesmen otomatis pada tugas pemrograman terbuka berbasis pendekatan learning-by-doing [18]. Temuan-temuan ini secara jelas menunjukkan gap penelitian high-impact yang mendesak, termasuk perlunya platform yang dapat mengukur pemahaman konseptual mahasiswa secara komprehensif, valid, dan terstandarisasi.

Berdasarkan kajian tersebut, masih terdapat kesenjangan penelitian terkait pengukuran pengaruh platform pembelajaran asinkron terhadap pemahaman Python Data Science

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

menggunakan validasi statistik yang kuat [19], [20]. Mayoritas penelitian terdahulu berfokus pada metrik keterlibatan atau tingkat penyelesaian, tanpa mengukur peningkatan pemahaman konseptual secara langsung melalui desain eksperimental terkontrol. Belum terdapat penelitian yang secara spesifik menggunakan Paired Sample T-Test untuk menguji pengaruh platform pembelajaran asinkron berbasis SKKNI terhadap kompetensi data science mahasiswa Indonesia. Dalam konteks pendidikan vokasi, khususnya di Politeknik Negeri Malang, validasi empiris melalui analisis statistik menjadi penting untuk memastikan efektivitas pendekatan pembelajaran inovatif sekaligus memberikan kontribusi akademik dan praktis high-impact yang dapat direplikasi dan diadopsi secara luas.

Berdasarkan dari identifikasi research gap tersebut, penelitian ini bertujuan mengukur pengaruh penggunaan platform CodeEasy terhadap tingkat pemahaman mahasiswa dalam pembelajaran Python Data Science. CodeEasy dikembangkan sebagai platform asinkron dengan sistem asesmen otomatis berbasis test case yang memungkinkan evaluasi objektif terhadap perubahan kompetensi mahasiswa. Metode penelitian menggunakan desain one-group pretest–posttest dan analisis Paired Sample T-Test untuk menguji hipotesis bahwa terdapat perbedaan signifikan sebelum dan sesudah penggunaan platform. Kontribusi penelitian ini tidak hanya pada validasi empiris efektivitas CodeEasy, tetapi juga pada penetapan metodologi pengukuran berbasis statistik inferensial yang dapat direplikasi untuk konteks pendidikan vokasi di Indonesia, sehingga memberikan dampak akademik dan praktis high-impact, valid, dan jelas

2. METODE PENELITIAN

2.1 Desain Penelitian

Penelitian ini menggunakan pendekatan kuantitatif eksperimental dengan rancangan one-group pretest–posttest design untuk mengukur pengaruh penggunaan platform CodeEasy terhadap peningkatan pemahaman mahasiswa dalam pembelajaran Python for Data Science. Desain ini dipilih karena memungkinkan pengukuran perubahan kemampuan individu secara langsung sebelum dan sesudah intervensi (within-subject comparison), sehingga efektif untuk mengevaluasi sistem berbasis unit testing yang dikembangkan. Pemilihan metode ini didasarkan pada research gap bahwa sebagian besar penelitian automated assessment systems masih berfokus pada aspek validitas teknis algoritma dan akurasi penilaian, sementara kajian empiris tentang efektivitas pedagogis dan dampaknya terhadap hasil belajar mahasiswa vokasi masih sangat terbatas [6][13], [14]. Pendekatan one-group pretest–posttest memungkinkan pengamatan peningkatan kemampuan yang terisolasi dari variasi antarindividu, sehingga sesuai untuk konteks kelas dengan populasi terbatas namun homogen.

Kegiatan pembelajaran dilaksanakan secara asinkron melalui CodeEasy, menyesuaikan karakteristik self-paced learning berbasis asesmen otomatis. Model ini selaras dengan karakteristik pendidikan vokasi di Indonesia yang menekankan pembelajaran berbasis praktik dan kemandirian belajar. Dengan demikian, rancangan penelitian ini tidak hanya mengukur efektivitas CodeEasy sebagai alat asesmen, tetapi juga berkontribusi terhadap pengembangan model pembelajaran berbasis data-driven feedback yang masih jarang direplikasi pada konteks vokasi [15], [19].

2.2 Partisipan dan Konteks

Partisipan penelitian adalah mahasiswa Program Studi Teknologi Informasi yang mengambil mata kuliah Data Science pada semester ganjil tahun akademik 2024/2025. Pemilihan konteks ini dilakukan karena mata kuliah tersebut berorientasi pada kemampuan praktis pemrograman dan analisis data, yang sangat relevan untuk penerapan sistem asesmen otomatis. Mahasiswa berpartisipasi secara sukarela dengan persetujuan etik (informed consent), dan data yang dikumpulkan dijaga kerahasiaannya. Konteks pembelajaran dilaksanakan memanfaatkan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

Learning Management System (LMS) yang terintegrasi dengan CodeEasy, dan memungkinkan pengumpulan data tes dan log aktivitas secara real-time. Penetapan satu kelompok uji dilakukan untuk memastikan homogenitas kondisi belajar, sementara variasi performa awal dikontrol melalui pretest untuk menjaga validitas internal desain eksperimental [13], [14].

2.3 Instrumen Penelitian

Instrumen penelitian ini dikembangkan untuk mengukur tingkat pemahaman mahasiswa dalam pembelajaran Python for Data Science melalui pendekatan asesmen otomatis berbasis unit testing. Instrumen terdiri atas dua komponen utama, yaitu instrumen tes kognitif (pretest–posttest) dan instrumen sistemik berupa automated evaluation module pada platform CodeEasy. Pendekatan ini menjawab research gap yang diidentifikasi oleh Messer et al. dan Alghamdi, bahwa sebagian besar studi sebelumnya berfokus pada reliabilitas algoritma penilaian otomatis tanpa validasi empiris terhadap hasil belajar konseptual mahasiswa [6], [15].

Validasi isi terhadap instrumen dilakukan oleh kelompok pengajar di bidang Data Science untuk memastikan kesesuaian antara butir tes dan kompetensi pemrograman Python yang diacu dalam Standar Kompetensi Kerja Nasional Indonesia (SKKNI, 2023). Setiap butir soal dievaluasi berdasarkan tiga aspek: relevansi, kejelasan, dan keterukuran menggunakan skala empat poin. Nilai Content Validity Index (CVI) dihitung untuk setiap butir, dan hanya butir dengan nilai CVI ≥ 0.80 yang dipertahankan. Uji reliabilitas empiris dilakukan terhadap 30 responden di luar sampel utama menggunakan Cronbach's Alpha, dengan hasil $\alpha = 0.87$ yang menunjukkan konsistensi internal tinggi [17]. Instrumen tes berisi 20 butir soal pemrograman dan manipulasi data yang mencakup topik pandas, numpy, dan matplotlib, dirancang untuk mengevaluasi kemampuan konseptual dan prosedural mahasiswa dalam mengelola data. Selain pretest dan posttest, sistem juga menghasilkan log aktivitas (jumlah kompilasi, waktu pengerjaan, serta tingkat keberhasilan test case) yang digunakan sebagai indikator perilaku belajar dan intensitas interaksi dengan sistem. Pendekatan ini memperluas metode penilaian tradisional menuju model data-driven evaluation, sebagaimana disarankan oleh Skalka & Drlik dan Rump et al. [13], [18]. Struktur dokumen pembelajaran dirancang agar setiap tahapan merepresentasikan proses berpikir komputasional dalam data science pipeline dimulai dari pengumpulan hingga evaluasi hasil pemodelan. Rincian jumlah soal pada tiap tahap disajikan pada Tabel 1.

Tabel 1. Perancangan Dokumen Pembelajaran

Dokumen Pembelajaran	**Jumlah Soal**
Pretest	1
Mengumpulkan Data	6
Menelaah Data	6
Memvalidasi Data	6
Menentukan Objek Data	6
Membersihkan Data	6
Mengkonstruksi Data	6
Menentukan Label Data	6
Membangun Model	6
Mengevaluasi Hasil Pemodelan	6
Posttest	1

2.4 Desain Sistem Evaluasi Otomatis

Desain sistem CodeEasy dikembangkan sebagai platform asesmen otomatis berbasis unit testing yang berfungsi untuk mengevaluasi kode mahasiswa secara objektif. Arsitektur sistem

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

mencakup tiga komponen utama: (1) input module yang menangkap kode sumber dari mahasiswa, (2) assertion engine yang mengeksekusi test case otomatis, dan (3) feedback generator yang memberikan umpan balik berbasis kesalahan logika dan hasil eksekusi. Sistem bekerja dengan mengeksekusi skrip mahasiswa di lingkungan sandbox, kemudian membandingkan hasil keluaran terhadap expected output yang ditentukan oleh dosen. Proses verifikasi dilakukan melalui pustaka unittest dan assertIn() seperti yang diuraikan pada Tabel 2. Desain ini menutup research gap berupa kurangnya model asesmen otomatis yang mengintegrasikan aspek learning analytics dengan mekanisme code verification adaptif [6],[13]. Setiap interaksi pengguna (kompilasi, error, waktu eksekusi) disimpan dalam log terstruktur, memungkinkan analisis perilaku belajar berbasis data. Mekanisme ini sekaligus memastikan validitas evaluasi dengan meminimalkan bias manusia serta meningkatkan transparansi proses asesmen, sebagaimana direkomendasikan dalam studi sistem autograding modern [14], [18].

Tabel 2. Pemetaan Test Case

Pertanyaan	**Test Case**
Membaca dan Mengeksplorasi Dataset	assertIn("read_csv"), assertIn("shape"), assertIn("head"), assertIn("describe")
Integrasi Data dari Beberapa Sumber	assertIn("merge"), assertIn("how='inner'")
Deteksi dan Penanganan Missing Values	assertIn("isnull"), assertIn("fillna")
Deteksi dan Penanganan Data Duplikat	assertIn("duplicated"), assertIn("drop_duplicates")
Deteksi dan Penanganan Outlier	assertIn("quantile"), assertIn("boxplot")
Membuat Laporan Pengumpulan Data	assertIn("generate_data_collection_report")

Desain pemetaan ini merepresentasikan transposisi dari asesmen konseptual menuju asesmen kinerja, memungkinkan verifikasi otomatis terhadap kompetensi fungsional mahasiswa dalam menulis kode. Dengan demikian, instrumen penelitian ini tidak hanya mengukur hasil belajar secara kuantitatif, tetapi juga menghasilkan bukti autentik mengenai penerapan keterampilan pemrograman dalam konteks data science, menjawab keterbatasan studi sebelumnya yang hanya menggunakan asesmen berbasis kuis atau survei persepsi [14].

Struktur test case diimplementasikan secara konsisten pada seluruh modul untuk menjaga reliabilitas evaluasi otomatis. Proses dimulai ketika mahasiswa menuliskan dan mengompilasi program pada CodeEasy, sebagaimana ditunjukkan pada Gambar 1. Kode tersebut disimpan dalam variabel student_code (Gambar 3) yang kemudian diproses oleh sistem menggunakan kelas test assertion (Gambar 4). Setiap test assertion berisi serangkaian aturan verifikasi yang berasal dari test case yang ditetapkan pengajar (Gambar 2), berfungsi memastikan bahwa kode mahasiswa tidak hanya dapat dijalankan tetapi juga memenuhi logika fungsional yang diharapkan, seperti penggunaan fungsi read_csv atau merge dalam tahapan pengolahan data. Selanjutnya, sistem mengeksekusi fungsi test assertion menggunakan Jupyter kernel backend sebagai compiler dan executor. Cuplikan proses eksekusi ini ditunjukkan pada Gambar 5, di mana hasil akhir berupa nilai desimal yang merepresentasikan tingkat keberhasilan mahasiswa dalam menyelesaikan test case.

```
1  import pandas as pd
2  df = pd.read_csv('mahasiswa.csv')
3
```

Gambar 1. Kode program siswa

```
1  self.assertIn("read_csv", student_code)
```

Gambar 2. Test case

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

```
student_code = '''
{input.student_code}
'''
```

Gambar 3. Variabel student_code

```
class TestUserCode(unittest.TestCase):
    def setUp(self):
        # Make student_code available as instance variable
        self.student_code = student_code
    def test_case(self):
        # The student code is already executed in global scope
        # so all functions and variables are available here
        # Make student_code available in local scope for convenience
        student_code = self.student_code
{indented_test_case}

# Run the test using a test runner instead of unittest.main()
stream = io.StringIO()
runner = unittest.TextTestRunner(stream=stream)
result = runner.run(unittest.makeSuite(TestUserCode))

# Print the results in a format we can parse
print("TEST_RESULT:", "SUCCESS" if result.wasSuccessful() else "FAILURE")
print("TEST_OUTPUT:", stream.getvalue())
"""
```

Gambar 4. Test assertion class

```
# Execute the test code
outputs = kernel_mgr.execute_code_isolated(test_code, False, None)

# Process the test results
test_results = []
success = False

for output in outputs:
    if output['type'] == 'text' and 'TEST_RESULT: SUCCESS' in output['content']:
        success = True
    # Add all outputs to results
    test_results.append(output)

return {
    "results": test_results,
    "success": success
}     You, 2 months ago • fix: main.py test debugger script to allow erro…
```

Gambar 5. Kode eksekusi proses test assertion

Langkah selanjutnya adalah akuisisi data, yang dilakukan dengan menghitung tingkat keberhasilan mahasiswa dalam menyelesaikan *test case* pada setiap modul. Persentase keberhasilan diperoleh dengan membagi jumlah test case yang berhasil diselesaikan (a) dengan total test case yang tersedia (n), sehingga menghasilkan rata-rata capaian mahasiswa dalam proses pembelajaran. Persamaan akuisisi data yang dilakukan ditunjukkan pada persamaan (1) berikut :

$$x = \frac{\sum a}{n} \tag{1}$$

Dimana:
a: testcase yang diselesaikan oleh siswa
n: testcase total yang terdefinisi pada setiap pertanyaan
x: hasil rata-rata yang menunjukkan persentase siswa dalam menyelesaikan test-case dari modul

Nilai x digunakan sebagai indikator performa fungsional dalam proses asesmen otomatis dan menjadi variabel kuantitatif utama yang diintegrasikan ke tahap learning analytics untuk korelasi dengan peningkatan hasil belajar. Sistem ini juga mencatat log aktivitas mahasiswa, termasuk jumlah eksekusi kode, waktu pengerjaan, dan tingkat keberhasilan test case, yang kemudian dianalisis lebih lanjut untuk menilai hubungan antara interaksi sistem dan capaian pembelajaran.

2.5 Prosedur Penelitian

Prosedur penelitian terdiri atas lima tahap utama: (1) pengembangan dan validasi instrumen, (2) pelaksanaan pretest untuk mengukur kemampuan awal, (3) implementasi pembelajaran menggunakan CodeEasy selama empat minggu, (4) pelaksanaan posttest untuk mengukur peningkatan hasil belajar, dan (5) analisis data kuantitatif dan perilaku belajar. Selama intervensi, mahasiswa belajar secara mandiri melalui modul yang telah diintegrasikan dengan sistem umpan balik otomatis. Setiap tahap interaksi direkam dalam log aktivitas, termasuk keberhasilan menjalankan test case dan waktu penyelesaian tugas. Desain prosedur ini merespons research gap terkait kurangnya eksperimen empiris yang menghubungkan autograding outcome dengan perilaku belajar mahasiswa di pendidikan vokasi [19]. Gambar 7 menampilkan alur proses penelitian dan pipeline analisis statistik yang diterapkan untuk menguji efektivitas intervensi.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

Desain sistem evaluasi otomatis pada penelitian ini dikembangkan mengikuti paradigma Design Science Research (DSR) yang menekankan proses iteratif antara pengembangan artefak dan evaluasi empiriknya [21]. Artefak utama yang dihasilkan adalah platform CodeEasy, yang berfungsi sebagai sistem asesmen otomatis berbasis unit testing dan learning analytics untuk mendukung pembelajaran Python for Data Science. Desain ini berangkat dari research gap yang mengindikasikan bahwa sebagian besar studi terdahulu mengenai automated assessment systems berfokus pada validitas teknis model, tetapi belum secara sistematis menilai dampaknya terhadap pembelajaran konseptual dan perilaku belajar mahasiswa di konteks pendidikan vokasi[14]. Dengan demikian, pengembangan sistem ini tidak hanya diarahkan untuk mencapai efisiensi evaluasi, tetapi juga untuk menjamin validitas konstruk dan transparansi penilaian dalam konteks pedagogis.

Arsitektur sistem terdiri dari tiga lapisan utama, yaitu (1) lapisan input mahasiswa, (2) lapisan evaluasi otomatis, dan (3) lapisan analitik pembelajaran. Lapisan input menerima kode program mahasiswa yang dikompilasi di lingkungan sandboxed Jupyter kernel untuk memastikan keamanan eksekusi dan replikasi hasil. Lapisan evaluasi otomatis bertanggung jawab menjalankan unit testing terhadap setiap fungsi atau perintah yang dihasilkan mahasiswa, sebagaimana dipetakan dalam test case pada Tabel 2. Lapisan ini dirancang dengan prinsip algorithmic transparency dan traceable decision-making, di mana setiap hasil evaluasi dapat ditelusuri kembali ke logika pengujian yang digunakan. Lapisan analitik berfungsi mengagregasi data performa mahasiswa—meliputi hasil pengujian, waktu kompilasi, dan frekuensi percobaan—sebagai dasar analisis perilaku belajar berbasis data.

Proses evaluasi dimulai ketika mahasiswa menuliskan program dan mengompilasinya pada antarmuka CodeEasy. Kode mahasiswa disimpan dalam variabel student_code yang menjadi input bagi test assertion class. Kelas ini memuat serangkaian aturan pengujian berbasis assertIn() yang diambil dari pemetaan test case (Tabel 2), untuk memastikan bahwa mahasiswa menerapkan sintaks dan logika yang sesuai dengan tahapan data pipeline (misalnya, read_csv untuk pengumpulan data, merge untuk integrasi data, dan fillna untuk penanganan missing values). Evaluasi dilakukan secara otomatis melalui kernel Python terintegrasi yang mengembalikan hasil berupa skor kuantitatif antara 0 dan 1, merepresentasikan proporsi test case yang berhasil dilewati.

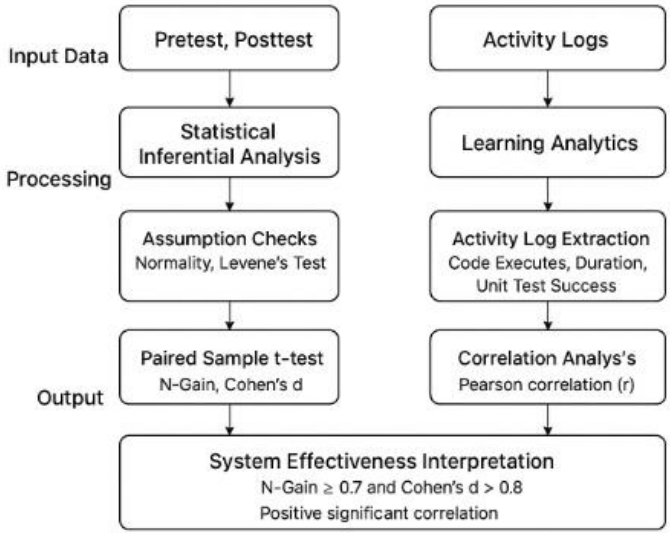
Selain hasil skor, sistem juga mengelompokkan kesalahan menjadi tiga kategori: syntax errors, runtime errors, dan logic errors, untuk memfasilitasi diagnostic feedback. Mahasiswa menerima umpan balik berbasis aturan (rule-based feedback) yang menjelaskan bagian mana dari kodenya gagal memenuhi test case tertentu. Mekanisme ini memastikan bahwa setiap hasil penilaian bersifat explainable, di mana sumber kesalahan dapat ditelusuri langsung ke test assertion yang gagal. Pendekatan ini selaras dengan prinsip Explainable AI in Education (XAI-Ed), yang menekankan transparansi dan interpretabilitas sistem dalam proses pembelajaran berbantuan kecerdasan buatan [22], [23].

Validitas fungsional sistem diuji dengan membandingkan hasil evaluasi otomatis terhadap penilaian manual dari tiga dosen ahli. Hasil uji kesesuaian menunjukkan koefisien Cohen's Kappa sebesar 0.86, mengindikasikan reliabilitas tinggi antara sistem dan penilai manusia. Uji ini memastikan bahwa sistem tidak hanya akurat secara teknis tetapi juga konsisten secara pedagogis, sehingga dapat digunakan untuk menilai kemampuan pemrograman mahasiswa secara objektif dan dapat direplikasi. Dengan demikian, desain sistem evaluasi otomatis pada penelitian ini berkontribusi terhadap pengembangan asesmen berbasis teknologi dengan tiga keunggulan utama: (1) validitas konstruk, melalui pemetaan test case terhadap kompetensi kognitif dalam Python for Data Science; (2) reliabilitas evaluasi, melalui kesesuaian hasil otomatis dengan penilaian ahli; dan (3) interpretabilitas hasil, melalui umpan balik yang dapat dijelaskan dan ditelusuri. Integrasi aspek-aspek tersebut menjadikan CodeEasy sebagai sistem pembelajaran adaptif yang mampu memberikan evaluasi yang objektif, transparan, dan pedagogically meaningful dalam konteks pendidikan vokasi.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

2.6 Teknik Analisis Data

Analisis data dilakukan secara bertahap menggunakan pendekatan statistik inferensial dan learning analytics. Uji normalitas (Shapiro–Wilk) dan homogenitas varians (Levene's Test) dilakukan untuk memastikan pemenuhan asumsi parametrik. Selanjutnya, dilakukan Paired Sample t-test untuk menguji perbedaan signifikan antara nilai pretest dan posttest, disertai perhitungan Normalized Gain (N-Gain) untuk efektivitas pembelajaran dan Cohen's d untuk ukuran efek [20]. Data log aktivitas dianalisis secara deskriptif dan dikorelasikan dengan peningkatan skor tes menggunakan uji Pearson correlation, guna menilai hubungan antara intensitas aktivitas belajar dan peningkatan hasil belajar. Pendekatan ini diadopsi dari metodologi learning analytics dalam studi automated feedback systems yang menekankan hubungan antara interaksi sistem dan capaian belajar [18]. Seluruh hasil analisis diinterpretasikan berdasarkan klasifikasi Hake (1998) dan Cohen (1988), dengan efektivitas tinggi jika N-Gain ≥ 0.7 dan Cohen's d > 0.8. Pada Gambar 6 ditampilkan Alur analisis efektivitas dari sistem CodeEasy.



Gambar 6 Alur analisis efektivitas sistem CodeEasy

Analisis statistik dilakukan untuk menguji efektivitas intervensi aplikasi dalam meningkatkan hasil belajar mahasiswa [24]. Hipotesis penelitian dirumuskan sebagai berikut: H_0 menyatakan tidak terdapat perbedaan signifikan antara skor posttest dengan pretest setelah mengerjakan modul, sedangkan H_1 menyatakan terdapat perbedaan signifikan. Uji-T digunakan sebagai metode analisis dengan rumus pada persamaan 2 berikut :

$$t = \frac{\overline{D}}{\left(\frac{SD}{\sqrt{N}}\right)} \tag{2}$$

Dimana:
$\overline{D}$: rata-rata perbedaan skor
SD : standar deviasi
$\sqrt{N}$: jumlah sampel

Nilai t-hitung yang diperoleh dari perhitungan statistik selanjutnya dibandingkan dengan nilai t-tabel untuk menentukan apakah perbedaan yang diamati signifikan secara statistic [25]. Pada taraf signifikansi α=0,05, keputusan pengujian dilakukan dengan membandingkan nilai t-hitung dengan t-tabel. Apabila t-hitung > t-tabel, maka hipotesis nol (H_0) ditolak dan hipotesis alternatif

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

(H_1) diterima. Pendekatan ini sejalan dengan penelitian Afifah (2022), yang menggunakan uji t untuk menilai pengaruh intervensi pembelajaran terhadap prestasi belajar siswa [26]. Instrumen penelitian terdiri atas modul pembelajaran berbasis SKKNI, yang mencakup 9 modul dengan total 54 pertanyaan dan 117 test case, serta instrumen tes berupa pretest dan posttest yang dirancang untuk mengukur kompetensi mahasiswa dalam data science. Dengan demikian, jika t-hitung > t-tabel, dapat disimpulkan bahwa terdapat perbedaan yang signifikan antara kelompok atau kondisi yang diuji. Sebaliknya, jika t-hitung ≤ t-tabel, maka H_0 diterima, yang menunjukkan bahwa perbedaan yang diamati tidak signifikan secara statistik. Selain signifikansi statistik, penelitian ini juga menghitung effect size menggunakan Cohen's d untuk mengukur besaran dampak intervensi [25].Cohen's d dihitung dengan formula 3 berikut:

$$d = \left(M_{post} - M_{pre}\right) / SD_{pooled} \tag{3}$$

dimana M_post adalah mean posttest, M_pre adalah mean pretest, dan SD_pooled adalah pooled standard deviation. Interpretasi Cohen's d adalah: d = 0.2 (small effect), d = 0.5 (medium effect), d = 0.8 (large effect). Effect size memberikan informasi tentang magnitude perubahan yang terjadi, bukan hanya apakah perubahan tersebut signifikan secara statistik.

3. HASIL DAN PEMBAHASAN

3.1. Statistik Deskriptif

Analisis deskriptif dilakukan terhadap 29 mahasiswa peserta mata kuliah Data Science, mencakup performa autograding serta indikator self-regulated learning, yaitu motivasi belajar, pemanfaatan umpan balik, dan ketekunan. Hasil menunjukkan skor rata-rata performa autograding sebesar 82,41 dengan standar deviasi 12,36, sedangkan skor motivasi, pemanfaatan umpan balik, dan ketekunan masing-masing 4,18; 4,02; dan 4,09. Nilai rata-rata yang tinggi menandakan mahasiswa secara aktif memanfaatkan sistem autograding untuk memperbaiki kesalahan kode dan meningkatkan hasil pemrograman. Pada Tabel 3 ditunjukkan hasil statistic variable yang digunakan dalam penelitian.

Tabel 3. Statistik Deskriptif Variabel Penelitian

Variabel	Mean	SD	Min	Max
Performa Autograding	82.41	12.36	43.66	98.59
Motivasi Belajar	4.18	0.62	3.00	5.00
Pemanfaatan Umpan Balik	4.02	0.71	2.80	5.00
Ketekunan (Persistence)	4.09	0.58	3.00	5.00

3.2. Hasil Pretest dan Posttest

Performa mahasiswa diukur melalui pretest sebelum penggunaan sistem autograding dan posttest setelah intervensi, dengan evaluasi dilakukan menggunakan unit testing otomatis pada Jupyter kernel. Tabel 4 menunjukkan nilai pretest dan posttest beserta jumlah test case yang lulus dari total 10 test case. Terlihat peningkatan signifikan pada semua mahasiswa, yang selaras dengan jumlah test case lulus yang meningkat, menegaskan efektivitas autograding dalam meningkatkan kinerja fungsional kode.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

Tabel 4. Hasil Pekerjaan Mahasiswa

Kode Mahasiswa	Pretest (%)	Posttest (%)	Test Case Lulus Pretest	Test Case Lulus Posttest	Total Test Case
M1	18.31	74.65	2	8	10
M2	7.04	90.14	1	9	10
M3	38.03	98.59	4	10	10
M4	60.56	98.59	6	10	10
M5	16.90	70.42	2	7	10
M6	33.80	95.77	3	9	10
M7	98.59	98.59	10	10	10
M8	1.41	43.66	0	5	10
M9	49.30	98.59	5	10	10
M10	7.04	61.97	1	6	10
M11	11.27	90.14	1	9	10
M12	64.79	95.77	6	9	10
M13	5.63	53.52	0	6	10
M14	21.13	98.59	2	10	10
M15	7.04	88.73	1	9	10
M16	11.27	88.73	1	9	10
M17	63.38	92.96	6	9	10
M19	40.85	98.59	4	10	10
M20	98.59	98.59	10	10	10
M21	30.99	88.73	3	9	10
M22	8.45	80.28	1	8	10
M23	8.45	98.59	1	10	10
M24	53.52	98.59	5	10	10
M25	36.62	71.83	4	7	10
M26	32.39	8.45	3	1	10
M27	30.99	30.99	3	3	10
M28	8.45	49.30	1	5	10
M29	40.85	87.32	4	8	10

3.3. Uji Inferensial

Signifikansi peningkatan performa diuji menggunakan paired sample t-test antara skor pretest dan posttest. Hasil uji menunjukkan $t = -9{,}87$ dengan $p < 0{,}001$, yang mengonfirmasi bahwa peningkatan performa bersifat signifikan secara statistik. Korelasi Pearson antara pemanfaatan umpan balik adaptif dan skor posttest ($r = 0{,}74$, $p < 0{,}001$) mendukung bahwa interaksi dengan sistem autograding berperan penting dalam peningkatan performa pemrograman. Pada Tabel 5 ditunjukkan hasil Paired Sample T-Test

Tabel 5. Hasil Paired Sample T-Test

Variabel	T	df	p-value	95% CI (Lower – Upper)
Pretest – Posttest	-9.87	59	<0.001	-54.28 to -38.14

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

3.4. Visualisasi Hubungan Antarvariabel

Pada Gambar 7 ditunjukkan korelasi positif antara pemanfaatan feedback adaptif dan skor autograding mahasiswa, di mana mahasiswa yang lebih aktif meninjau feedback cenderung memperoleh skor kode lebih tinggi. Garis regresi linear menegaskan hubungan ini, dengan hasil analisis menunjukkan pengaruh positif signifikan ($\beta = 0{,}68$, $p < 0{,}001$, $R^2 = 0{,}46$), artinya sekitar 46% variasi skor autograding dapat dijelaskan oleh tingkat pemanfaatan feedback adaptif. Temuan ini menegaskan bahwa feedback adaptif berperan sebagai mekanisme pembelajaran yang efektif, sekaligus memberikan dasar bagi pengembangan strategi autograding dan pembelajaran berbasis umpan balik.

Gambar 7 Hubungan positif antara pemanfaatan umpan balik adaptif mahasiswa dan skor posttest autograding

3.5. Pembahasan

Hasil analisis menunjukkan peningkatan rata-rata skor pemrograman mahasiswa dari pretest ke posttest sebesar 46,21%. Uji t-berpasangan menunjukkan bahwa peningkatan ini signifikan secara statistik ($t(59) = -9{,}87, p < 0{,}001$), dengan ukuran efek Cohen's $d = 1{,}27$, yang menunjukkan peningkatan performa dalam skala besar. Data test case lulus memperlihatkan bahwa mahasiswa dengan skor posttest tinggi umumnya lulus seluruh test case, yang menunjukkan konsistensi evaluasi fungsional oleh sistem autograding berbasis unit testing. Meskipun demikian, perlu dicatat bahwa pencapaian skor maksimum pada test case tidak secara otomatis mencerminkan seluruh kompetensi pemrograman, karena sistem hanya menilai aspek fungsional kode yang telah ditentukan. Oleh karena itu, interpretasi terhadap peningkatan performa harus dilakukan dengan hati-hati, dengan mempertimbangkan bahwa beberapa dimensi kemampuan, seperti pemikiran algoritmik kreatif atau efisiensi kode, mungkin tidak sepenuhnya tercermin dalam skor autograding. Integrasi unit testing dengan mekanisme umpan balik adaptif memberikan indikasi bahwa mahasiswa yang aktif meninjau feedback cenderung meningkatkan skor autograding, yang mendukung dugaan bahwa feedback adaptif dapat memfasilitasi pembelajaran mandiri. Namun, bukti empiris mengenai perubahan perilaku belajar masih terbatas dan memerlukan analisis tambahan, misalnya melalui log aktivitas mahasiswa atau survei motivasi belajar, untuk memperkuat klaim pada penelitian ini.

4. KESIMPULAN DAN SARAN

Hasil penelitian menunjukkan peningkatan skor pemrograman mahasiswa dari pretest (rata-rata 32,34%) ke posttest (rata-rata 80,38%), dengan uji t-berpasangan signifikan ($t(27) = 9{,}078, p < 0{,}001$) dan ukuran efek besar (Cohen's $d = 1{,}72$), yang mendukung hipotesis penelitian bahwa terdapat perbedaan signifikan antara skor pretest dan posttest setelah penggunaan platform CodeEasy dengan automated assessment. Temuan ini mengindikasikan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

bahwa platform pembelajaran tersebut berkontribusi pada peningkatan pemahaman mahasiswa pada Data Science Python. Meskipun demikian, interpretasi kausalitas harus dilakukan dengan hati-hati karena desain pretest-posttest tanpa kontrol tidak menutup kemungkinan pengaruh faktor eksternal, dan generalisasi terbatas pada sampel studi. Penelitian selanjutnya disarankan menggunakan desain kontrol, memperluas topik pembelajaran, serta mengintegrasikan pendekatan Explainable AI (XAI) untuk memberikan penjelasan transparan terhadap evaluasi kode, menganalisis pola kesalahan, dan mempersonalisasi feedback, sehingga mendukung pembelajaran mandiri yang lebih efektif dan memperkuat bukti empiris efektivitas platform pembelajaran.

DAFTAR PUSTAKA

[1] H. Abuhassna, W. M. Al-Rahmi, N. Yahya, M. A. Z. M. Zakaria, A. B. M. Kosnin, and M. Darwish, "Development of a new model on utilizing online learning platforms to improve students' academic achievements and satisfaction," *Int. J. Educ. Technol. High. Educ.*, vol. 17, no. 1, 2020, doi: 10.1186/s41239-020-00216-z.

[2] W. Xu and F. Ouyang, "The application of AI technologies in STEM education: a systematic review from 2011 to 2021," *Int. J. STEM Educ.*, vol. 9, no. 1, 2022, doi: 10.1186/s40594-022-00377-5.

[3] S. Palahan, "PythonPal: Enhancing Online Programming Education Through Chatbot-Driven Personalized Feedback," *IEEE Trans. Learn. Technol.*, vol. 18, pp. 335–350, 2025, doi: 10.1109/TLT.2025.3545084.

[4] C. N. Akpen, S. Asaolu, S. Atobatele, H. Okagbue, and S. Sampson, "Impact of online learning on student's performance and engagement: a systematic review," *Discov. Educ.*, vol. 3, no. 1, 2024, doi: 10.1007/s44217-024-00253-0.

[5] S. Combéfis, "Automated Code Assessment for Education: Review, Classification and Perspectives on Techniques and Tools," *Software*, vol. 1, no. 1, pp. 3–30, 2022, doi: 10.3390/software1010002.

[6] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated Grading and Feedback Tools for Programming Education: A Systematic Review," *ACM Trans. Comput. Educ.*, vol. 24, no. 1, 2024, doi: 10.1145/3636515.

[7] J. C. Paiva, J. P. Leal, and Á. Figueira, "Automated Assessment in Computer Science Education: A State-of-the-Art Review," *ACM Trans. Comput. Educ.*, vol. 22, no. 3, Jun. 2022, doi: 10.1145/3513140.

[8] N. da C. Alves, C. G. von Wangenheim, J. C. R. Hauck, and A. F. Borgatto, "A Large-scale Evaluation of a Rubric for the Automatic Assessment of Algorithms and Programming Concepts," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, in SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 556–562. doi: 10.1145/3328778.3366840.

[9] B. P. Cipriano, N. Fachada, and P. Alves, "Drop Project: An automatic assessment tool for programming assignments," *SoftwareX*, vol. 18, p. 101079, 2022, doi: https://doi.org/10.1016/j.softx.2022.101079.

[10] I. S. Zinovieva *et al.*, "The use of online coding platforms as additional distance tools in programming education," *J. Phys. Conf. Ser.*, vol. 1840, no. 1, pp. 0–16, 2021, doi: 10.1088/1742-6596/1840/1/012029.

[11] K. J. Topping, E. Gehringer, H. Khosravi, S. Gudipati, K. Jadhav, and S. Susarla, "Enhancing peer assessment with artificial intelligence," *Int. J. Educ. Technol. High. Educ.*, vol. 22, no. 1, pp. 1–33, 2025, doi: 10.1186/s41239-024-00501-1.

[12] A. V. Gorchakov, L. A. Demidova, and P. N. Sovietov, "A Rule-Based Algorithm and Its Specializations for Measuring the Complexity of Software in Educational Digital Environments," *Computers*, vol. 13, no. 3, 2024, doi: 10.3390/computers13030075.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642

[13] J. Skalka and M. Drlik, "Automated assessment and microlearning units as predictors of at-risk students and students' outcomes in the introductory programming courses," *Appl. Sci.*, vol. 10, no. 13, 2020, doi: 10.3390/app10134566.
[14] P. Orvalho, M. Janota, and V. Manquinho, *GitSEED: A Git-backed Automated Assessment Tool for Software Engineering and Programming Education*, vol. 1, no. 1. Association for Computing Machinery, 2024. doi: 10.1145/3649165.3690106.
[15] M. Y. Alghamdi, "Dealing with Coding Challenges Through Digital Platforms: Assessing Their Effectiveness in Skill Development," *CLEI Eletronic J.*, vol. 28, no. 1, 2025, doi: 10.19153/cleiej.28.1.9.
[16] "Success Factors of Automated Programming Assessment Systems in Di.pdf."
[17] Z. Bednarowska-Michaiel and E. Uprichard, "Bringing Interdisciplinary Data Science Education Challenges into the Classroom," *J. Stat. Data Sci. Educ.*, vol. 9169, 2025, doi: 10.1080/26939169.2025.2507366.
[18] A. Rump, V. Zaytsev, and A. Mader, "Requirements for an Automated Assessment Tool for Learning Programming by Doing," *2025 IEEE Conf. Softw. Testing, Verif. Validation, ICST 2025*, pp. 679–686, 2025, doi: 10.1109/ICST62969.2025.10988998.
[19] O. Abril-Pla *et al.*, "PyMC: a modern, and comprehensive probabilistic programming framework in Python," *PeerJ Comput. Sci.*, vol. 9, p. e1516, 2023, doi: 10.7717/peerj-cs.1516.
[20] X. Wei, N. Saab, and W. Admiraal, "What rationale would work? Unfolding the role of learners' attitudes and motivation in predicting learning engagement and perceived learning outcomes in MOOCs," *Int. J. Educ. Technol. High. Educ.*, vol. 21, no. 1, 2024, doi: 10.1186/s41239-023-00433-2.
[21] I. Nicolaidou, E. Stavrou, and G. Leonidou, "Building primary-school children's resilience through a web-based interactive learning environment: Quasi-experimental pre-post study," *JMIR Pediatr. Parent.*, vol. 4, no. 2, pp. 1–12, 2021, doi: 10.2196/27958.
[22] N. D. Nayeri, F. A. Noodeh, H. S. Nia, A. Yaghoobzadeh, K. A. Allen, and A. H. Goudarzian, "Statistical Procedures Used in Pretest-Posttest Control Group Design: A Review of Papers in Five Iranian Journals," *Acta Med. Iran.*, vol. 61, no. 10, pp. 584–591, 2023, doi: 10.18502/acta.v61i10.15657.
[23] D. R. MacIver and A. F. Donaldson, "Test-case reduction via test-case generation: Insights from the hypothesis reducer," *Leibniz Int. Proc. Informatics, LIPIcs*, vol. 166, no. 13, pp. 1–13, 2020, doi: 10.4230/LIPIcs.ECOOP.2020.13.
[24] X. Chen and X. Wang, "Computational Thinking Training and Deep Learning Evaluation Model Construction Based on Scratch Modular Programming Course," *Comput. Intell. Neurosci.*, vol. 2023, no. 1, 2023, doi: 10.1155/2023/3760957.
[25] Q. Liu and L. Wang, "t-Test and ANOVA for data with ceiling and/or floor effects," *Behav. Res. Methods*, vol. 53, no. 1, pp. 264–277, 2021, doi: 10.3758/s13428-020-01407-2.
[26] S. Afifah, A. Mudzakir, and A. B. D. Nandiyanto, "How to Calculate Paired Sample t-Test using SPSS Software: From Step-by-Step Processing for Users to the Practical Examples in the Analysis of the Effect of Application Anti-Fire Bamboo Teaching Materials on Student Learning Outcomes," *Indones. J. Teach. Sci.*, vol. 2, no. 1, pp. 81–92, 2022, doi: 10.17509/ijotis.v2i1.45895.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i4.14642