

Software Engineering Stage 6 (Year 12) – Sample Software Engineering Systems Report Template



Contents

1. Identifying and defining	2
1.1. Define and analyse problem requirements	2
1.2. Tools to develop ideas and generate solutions	3
2. Research and planning	4
2.1. Project management	4
2.2. Quality assurance	5
2.3. Systems modelling	6
3. Producing and implementing	12
4. Testing and evaluating	13
4.1. Evaluation of code	13
4.2. Evaluation of solution	14

1. Identifying and defining

1.1. Define and analyse problem requirements

Problem context

Students **analyse** the problem by **describing** each of its individual components and **explaining** how each of these components contribute to the problem needing resolution.

To train a machine learning model to identify a user on the Internet, we need a dataset containing sequences of accessed web pages. Exploratory data analysis (EDA) is performed, using insights to generate initial hypotheses. The dataset needs to be cleaned by removing any outliers, duplicates or erroneous data. Feature engineering adds additional features from existing ones, guided by hypotheses formed in the EDA step. In the training step, models are trained on a time-aware cross-validation scheme. The loss function is the average Receiver Operating Characteristic Area Under the Curve (ROC AUC) score. Models are chosen based on the baseline performance without hyperparameter tuning. Promising models are retained for hyperparameter optimization. In the evaluation step, the model is tested on the test data. Predictions are generated to create a ROC curve and precision-recall curves, showing the various metrics of classification. Potential improvements are implemented in sprints resulting in better models over time.

Needs and opportunities

Students **describe** the needs of the new system to be built based on the problem context and using the table given below.

Need	Description
1. Generalises well to unseen data	Able to perform well (>95% ROC AUC) in both training and testing data
2. Interpretability	Able to explain the predictions given by the model based on feature importances and algorithm interpretability.
3. Continuous improvement	Able to improve over time with additional data.

Boundaries

Students **analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc.

Computational cost is of particular concern to machine learning models. While machine learning models for classification such as support vector machine, nearest neighbor or decision trees can be run on any machines with Python 3 installed, deep learning classification models such as long short-term memory or recurrent neural networks require more computation power with dedicated GPUs.

1.3. Implementation method

Students **explain** the applicability of the implementation method for the current project.

These methods are normally direct, phased, parallel, or pilot.

Using an Object-Oriented Programming (OOP) approach, the project can be bundled into a complete pipeline from data to predictions. As a result, a direct implementation can be used for integration into existing intruder detection systems.

1.4. Financial feasibility

Students are to **conduct** a financial feasibility study, including producing an opening-day balance sheet, to assess whether their application is financially viable.

SWOT Analysis

<p>Strengths</p> <p>Lower false positive rate (5%) compared to industry average (14%)</p> <p>Low inference time of 15-50 milliseconds</p> <p>Based on behavioural analysis instead of traditional static approaches</p>	<p>Weaknesses</p> <p>Expensive hardware for more complex models</p> <p>Higher inference time (1-2 minutes) for deep learning models.</p> <p>Quality of the data dictate the performance of the model</p>
<p>Opportunities</p> <p>Increasing sophistication of cyber attacks creates demand for advanced solutions</p> <p>Traditional detection methods demonstrably insufficient for modern attacks</p> <p>Government initiatives promoting adoption of advanced security technologies</p>	<p>Threats</p> <p>Large security vendors investing heavily in AI-based solutions such as Palo Alto Network or McAfee</p> <p>Cybersecurity field evolving quickly with new approaches and technologies</p> <p>Sophisticated attackers may develop techniques to evade behavioral analysis</p>

Study	Go or No Go?	Assessment and evidence
Market feasibility	Go	The Intrusion Detection System (IDS) Market Size was valued at USD 5.71 Billion in 2023 and is expected to reach USD 11.43 Billion by 2032 and grow at a CAGR of 8.0% over the forecast period 2024-2032. (SNS Insider, 2024)

Study	Go or No Go?	Assessment and evidence
		<p>This shows the demand for such systems remains high and likely to grow at a rapid pace over time.</p>
Cost of development	Go	<p>Cloud services costs range from \$2000-10000 per month, reflected as a projection of \$36000 in the first year on the balance sheet.</p> <p>Labour costs are approximated at \$49/hour for a full-time software engineer, resulting in a \$97500 salary payable in the balance sheet.</p>
Cost of ownership	Go	<p>Initial capital requirements of AUD \$1.16M in 2024 are manageable with secured seed funding of \$450000 plus venture capital loans of \$525000.</p> <p>Initial deployment setup costs approximately \$15000-25000 per enterprise client including system integration, customization, and staff training.</p> <p>Cloud-based Software as a Service (SaaS) model eliminates on-premises hardware requirements for most clients, while API-first design enables automated deployment pipelines.</p>
Income potential	Go	<p>The cloud-based architecture allows cost-efficient scaling as revenue grows, with break-even achievable at ~\$50000 monthly operational costs. This is equivalent to 5 - 20 enterprise clients per month.</p>

Study	Go or No Go?	Assessment and evidence
		Gross revenue of 891 million USD can be achieved with a conservative market penetration rate of 5%.
Future expansion opportunities	Go	<p>The behavioral analysis technology is applicable to mobile app security, IoT device monitoring and cloud workload protection, creating multiple revenue streams.</p> <p>Integration opportunities with major security platforms (Splunk, IBM QRadar) offer revenue sharing potential.</p>

Opening Day Balance Sheet

2. Research and planning

2.1. Project management

Software development approach

Students **explain** the software development approach most applicable for this current project. These are normally: Waterfall, Agile and WAgile.

An agile software development approach is most appropriate for machine learning tasks such as this one. This is due to the dynamic nature of such tasks and the need to adapt to changes based on iterative sprints. Each sprint is designed to implement a potential improvement to the model, evaluating the effectiveness of such improvements and weaknesses of the model to identify subsequent areas of improvement. Additionally, the complexity of machine learning tasks means that there are many unknowns at the time of

design. Agile approaches such as Scrum offers the flexibility needed for rapid planning changes and adaptability to new information as it emerges.

2.2. Quality assurance

Quality criteria

Students **explain** quality criteria based upon the needs from Section 1.1. These quality criteria should contain qualities, characteristics or components that need to be included or visible – based on Section 1.1. – by the end of the current project.

Quality criteria	Explanation
1. Generalises well to unseen data	Able to perform well (>95% ROC AUC) on out-of-sample data.
2. Interpretability	Able to explain the predictions given by the model based on feature importances and algorithm interpretability.
3. Continuous improvement	Able to improve over time with additional data.

Compliance and legislative requirements

Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian [NSW Privacy and Personal Information Act \(1998\)](#) and/or [Federal Privacy Act \(1988\)](#). Alternatively, international standards on information security management such as [ISO/IEC 27001](#) may also be applicable.

Compliance or legislative issue	Methods for mitigation
Privacy Act 1988 (Cth)	Comprehensive and transparent privacy policy.

Compliance or legislative issue	Methods for mitigation
	<p>Use data anonymization and pseudonymisation techniques.</p> <p>Implement purpose limitation to only intruder detection.</p>
Privacy and Personal Information Protection Act 1988 (NSW)	<p>Implement data minimization practices.</p> <p>Create separate data handling procedures for NSW public sector information.</p> <p>Ensure compliance with Information Protection Principles.</p>
ISO/IEC 27001	<p>Implement Information Security Management System.</p> <p>Conduct regular risk assessments and security audits.</p> <p>Create incident response and business continuity procedures</p> <p>Implement security awareness training programs</p>

2.3. Systems modelling

Algorithm design

Students **develop** algorithms using methods such as pseudocode or flowcharts to solve the problem and meet the needs from Section 1.1. These algorithms should explicitly include the variables from the data dictionaries created in the previous section.

Included in external documentation.

3. Producing and implementing

Solution to software problem

Students are to **include** screenshots of their final developed solution here. Each screenshot should include a caption that **explains** how it links to the:

- Needs identified in Section 1.1.
- Components of Section 2.3. such as storyboards, data dictionaries and so on.

Analyse class imbalances

```
plt.figure(figsize=(6, 4))
#New instance of train_df as we removed the target earlier.
sns.countplot(x='target', data=pd.read_csv(os.path.join(PATH_TO_DATA, 'train_sessions.csv')))
plt.title('Class Imbalance: Alice vs Intruder')
plt.xlabel('Class (0 = Intruder, 1 = Alice)')
plt.ylabel('Number of Sessions')
plt.xticks([0, 1], ['Intruder', 'Alice'])
plt.show()
```

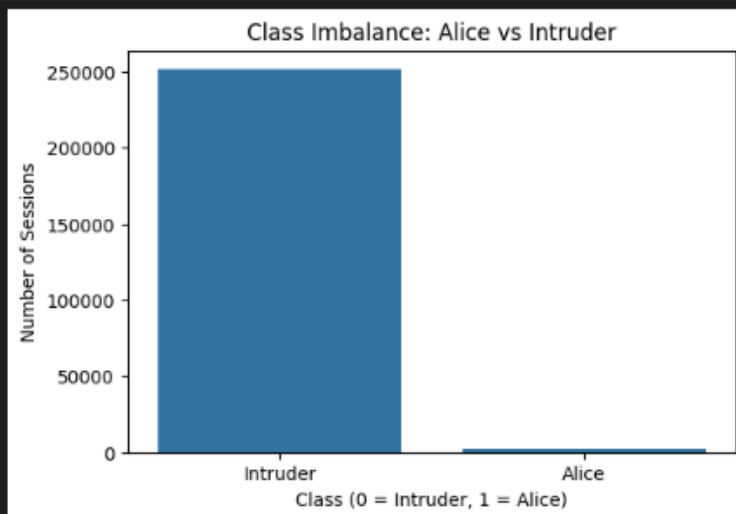


Figure 1. Sample graph produced in EDA

```
no_vectorizer_pipeline = FeatureUnion(transformer_list=[  
    ('feature_pipeline', feature_pipeline),  
    ('scaled_pipeline', scaled_pipeline)  
])
```

Figure 2. FeatureUnion containing Pipelines - modularized transformation steps in wrangling phase

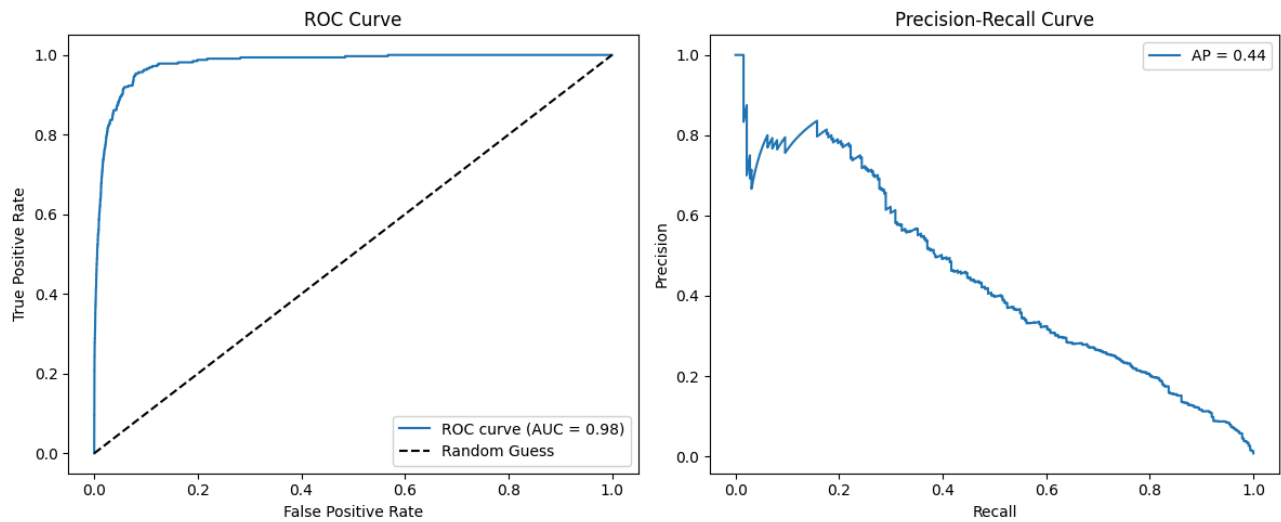


Figure 3. Graphs to visualize the performance of model training in the development phase (Final model).

Weight ²	Feature
+6.985	youwatch.org
+6.980	fr.leepwiki.com
+4.816	sk.com
+4.584	www.mls-journal.net
+4.531	www.hungar-china.fr
+4.501	www.madnessrights.net
+4.133	www.melly.fr
+4.112	r1--en-gn5uag-jqbe.googlevideo.com
+3.999	api.bing.com
+3.992	media-1.melly.fr
+3.826	dub119.mail.live.com
+3.715	fr.manu.com
+3.713	reviewer.livoclanord.fr
+3.707	r1--en-gn5uag-jqbe.googlevideo.com
+3.682	cid-edfc3ef65c6608a4.users.storage.live.com
+3.668	www35.glam.com
+3.643	s.videostep.com
+3.643	r3--en-gn5uag-jqbe.googlevideo.com
+3.631	glee.hypnoweb.net
+3.568	www.dailymotion.com
+3.554	il.yimg.com
+3.537	gg.google.com
+3.456	www.purepeople.com
+3.439	www.bbc.co.uk
+3.413	www.demotivateur.fr
+3.385	www.gn.justice.gouv.fr
... 4902 more positive ...	
... 4902 more negative ...	
-3.236	www.bing.com
-4.431	mail.google.com
-5.334	plus.google.com
-5.614	<BIAS>

Figure 4. Feature weights for model explainability

Version control

Students **describe** what version control system or protocol was implemented.

For version control, Git (through GitHub) was used to manage and track changes throughout the development lifecycle. Within the repository, branches were used for feature development, with commit history and pull requests for code reviews.

For software versioning, semantic versioning (Major.minor.patch) was used. Each new minor feature is implemented on a separate branch, so that patches can be made and tracked on that branch. This approach ensured stable releases, easier rollback, and parallel development of features and fixes without disrupting the main codebase.

4. Testing and evaluating

4.1. Evaluation of code

Methodology to test and evaluate code

Students **explain** the methodologies used to test and evaluate code. Methodologies include:

- Unit, subsystem and system testing

Unit testing is the process of testing the smallest functional component of code. Software testing helps ensure code quality and is an integral part of software development.

System testing is the process by which a software engineer evaluates how the various components of an application interact together to form a full, integrated system or application.

- Black, white and grey box testing

Black box testing is a technique used to test software solutions where the inputs and expected outputs are known. However, the processing/implementation occurring are unknown.

White box testing is a technique used to test software solutions where explicit knowledge of the internal workings of the item being tested is used. Also referred to as structural or open box testing.

Grey box testing is a technique used to test software solutions, in which the tester has some knowledge of the implementation of the component being tested.

- Quality assurance.

Quality assurance is a holistic management system aiming to maintain a desired level of quality at every stage of the software engineering process.

Code optimisation

Students **explain** the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

- Dead code elimination

Dead code elimination is the process of identifying and removing dead code - code that is never executed during runtime. It is essential for software efficiency, reducing complexity and improving maintainability. In my project, this is most visible in the deletion of unused imported modules.

- Code movement

Code movement (or code motion) is the process of identifying and relocating statements or computations that are prone to repeated computations. It is essential for software efficiency and reducing complexity. In my project, the most common type of code movement is Loop-Invariant Code Motion, particularly in the hyperparameter optimization process, where certain hyperparameters (e.g. solvers or penalty) are known to be most applicable and do not have to be optimized.

- Strength reduction

In strength reduction, expensive operations like multiplication and division are replaced by cheaper operations like bit shifting, addition or subtraction.

- Common sub-expression elimination

Common sub-expression elimination is the elimination of expressions or sub-expressions that have appeared and computed before and appear again during the computation of the code.

- Compile time evaluation – constant folding and constant propagation

Constant expressions are evaluated at compile time, exhaustively until no more expressions can be simplified. For example:

```
int a = 30;
int b = 9 - (a / 5);
int c = b * 4;

if (c > 10) {
    c = c - 10;
}
return c * (60 / a);
```

The compiler will evaluate a, then b, then c, at compile time.

- Refactoring

Refactoring is the process of restructuring the source code without changing its external behaviour. In my project, refactoring can be seen in reducing the number of `pandas.DataFrame.apply()`, which loops over every single row of a n-dimensional array, calling a Python function every time. The refactored code implements a better way of transforming the array, by using vectorization - using optimized, pre-compiled code written in a low-level language (e.g. C) to perform mathematical operations over a sequence of data.

```
def transform(self, X, y=None):
    hour = X['time1'].apply(lambda ts: ts.hour)
    period1 = ((hour >= 12) & (hour <= 13)) | ((hour >= 18) & (hour <= 19)).astype('int')
    period2 = ((hour >= 16) & (hour <= 18)).astype('int')
    period3 = ((hour <= 12) & (hour >= 0)) | ((hour >= 19) & (hour <= 24)) | ((hour >= 14) & (hour <= 15)).astype('int')
    month = X['time1'].apply(lambda ts: ts.month)
    peak_alice_months = ((month == 11) | (month == 2) | (month == 3)).astype('int')
    weekday = X['time1'].apply(lambda ts: ts.weekday()).astype('int')
    mon_tue = ((weekday == 0) | (weekday == 1)).astype('int')
    wed_sat_sun = ((weekday == 2) | (weekday == 5) | (weekday == 6)).astype('int')
    year = X['time1'].apply(lambda ts: ts.year).astype('int')
    X = np.c_[period1.values, period2.values, period3.values, peak_alice_months.values,
              mon_tue.values, wed_sat_sun.values]
    return X
```

```
%timeit no_vectorizer_pipeline.fit_transform(train_df)
✓ 23.1s
2.9 s ± 143 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Before refactoring

```
def transform(self, X, y=None):
    time1 = X['time1']
    hour = time1.dt.hour
    period1 = (((hour >= 12) & (hour <= 13)) | ((hour >= 18) & (hour <= 19))).astype(int)
    period2 = ((hour >= 16) & (hour <= 18)).astype(int)
    period3 = (((hour >= 0) & (hour <= 12)) | ((hour >= 14) & (hour <= 15)) | ((hour >= 19) & (hour <= 24))).astype(int)
    month = time1.dt.month
    peak_alice_months = ((month == 11) | (month == 2) | (month == 3)).astype(int)
    weekday = time1.dt.weekday
    mon_tue = ((weekday == 0) | (weekday == 1)).astype(int)
    wed_sat_sun = ((weekday == 2) | (weekday == 5) | (weekday == 6)).astype(int)
    # year = time1.dt.year.astype(int) # Not returned in output array
    X_new = np.c_[
        period1.values, period2.values, period3.values,
        peak_alice_months.values, mon_tue.values, wed_sat_sun.values
    ]
    return X_new
```

```
%timeit no_vectorizer_pipeline.fit_transform(train_df)
✓ 10.9s
1.37 s ± 54.7 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

After refactoring

4.2. Evaluation of solution











Analysis of feedback

Catch Me If You Can ("Alice")

Submit Prediction

...

OverviewDataCodeModelsDiscussionLeaderboardRulesTeamSubmissions

	baseline_logreg_submission (1).csv Complete · 17s ago	0.91806	<input checked="" type="checkbox"/>
	baseline_logreg_submission.csv Complete · 2m ago	0.90783	<input type="checkbox"/>
	best_params_xgb_submission.csv Complete · 2d ago	0.75605	<input type="checkbox"/>
	best_params_logreg_submission (5).csv Complete · 2d ago	0.87156	<input type="checkbox"/>
	best_params_logreg_submission (4).csv Complete · 2d ago	0.88658	<input type="checkbox"/>
	best_params_logreg_submission (3).csv Complete · 2d ago	0.89015	<input type="checkbox"/>
	best_params_logreg_submission (2).csv Complete · 2d ago	0.88628	<input type="checkbox"/>
	best_params_logreg_submission (1).csv Complete · 2d ago	0.80317	<input type="checkbox"/>
	best_params_logreg_submission.csv Complete · 2d ago	0.78721	<input type="checkbox"/>
	baseline_logreg_submission.csv Complete · 3d ago	0.91516	<input checked="" type="checkbox"/>

1. First feedback: baseline_logreg_submisison with a leaderboard score of 0.91516.
 - The first model is based on using keywords (URLs in this case) frequency in a text/corpus - a commonly used approach towards machine learning projects involving text analysis.
 - From this feedback, the model should benefit from more features involving time, given along with the URLs.
2. Second feedback and third feedback: best_params_logreg_submission and (1)
 - Hyperparameter optimization is done on a LogisticRegressor.
 - Engineered features from temporal data most likely added noise to the model, leading to the deterioration of leaderboard score.
3. 4th feedbacks: best_params_logreg_submission (2)

- Intraday features (morning, night) features were divided into periods and grouped, using insights from EDA, by different patterns in Alice's and intruders' behaviours.
 - This led to a slight improvement over the last iteration, but still lower than the baseline.
4. 5th - 7th feedbacks: `best_params_logreg_submission` (3) to (5)
- Features with low weights were removed from the dataset. Consequently, some noise was removed from the model.
 - This led to a slight improvement from the 4th feedback in the 5th submission to 0.89015 but still lower than baseline.
5. 8th feedback: `best_params_xgb_submission`
- Because the dataset has severe class imbalance, the "scale_pos_weight" hyperparameter of `XGBClassifier` might help mitigate this. `XGBClassifier` is also a more complex model, which might help with underfitting.
 - Despite an improvement in validation ROC AUC and average precision, the model did not perform well on the test set.
6. 9th and 10th feedback: `baseline_logreg_submission` and (1)
- Minor changes to one hyperparameter, solver, of the baseline model led to a minor improvement.

Testing methods

Students **identify** the method or methods of testing used in this current project. For each they use, students are to **explain** how and why it was used.

Method	Applicability	Reasoning
Functional testing	Yes	To verify that the machine learning pipeline runs as expected, e.g., model loads, processes input, and returns outputs without errors.
User Acceptance testing	No	No user-facing component, therefore no interface or user workflow to validate.

Method	Applicability	Reasoning
Live data	Yes	Model performance was tested using real-world data to evaluate accuracy and generalization.
Simulated data	Yes	Used to validate edge cases or test model behavior under controlled or rare scenarios.
Beta testing	No	No end-user involved. The model runs in a backend environment without external testers.
Volume testing	Yes	Ensures that the system can handle large datasets during training or inference without degradation or crashing.

Security Assessment

Students are to **perform** an extensive security assessment of their final application and **explain** the countermeasures implemented.

Threat	Countermeasure
Data leakage	All sensitive data was anonymized or removed. Only timestamps and websites are recorded/used for model development.
Model inversion attacks	Limited access to the model's output details. Only final predicted probabilities are submitted.
Insecure dependencies or libraries	Only well-known and stable libraries were used to minimize risk. These libraries are actively maintained and have strong community support and oversight.

Test data tables

Students **identify** variables which were used for either path and/or boundary testing. Students **develop** these test data tables based on their algorithms versus their real code. Students then **state** the reason for including said variables.

Variable	Maximum	Minimum	Default Value	Expected Output	Actual Output	Reason for Inclusion
Number of sites visited (siteID columns)	10	1	N/A	Model makes prediction with fewer or full inputs	Same as expected	To verify that the model handles sessions of varying lengths.
Session duration	1800	1	N/A	Model processes without failure	Same as expected	Enforces the 30-minute session limit.
SiteID	48371	1	N/A	Valid mapping and feature extraction	Same as expected	Validate boundary values from <code>site_dic.pkl</code>

Variable	Maximum	Minimum	Default Value	Expected Output	Actual Output	Reason for Inclusion
timeN (N from 1 to 10)	2262-04-11 23:47:16.854 775807	2013-00-00 00:00:00	N/A	Model handles missing timestamps	Same as expected	To verify that the model handles sessions of varying lengths.

Analysis of solution against quality success criteria

Students are to take each quality success criteria from Section 2.2 and place it here. For each quality criteria, **analyse** the components of the solution that met or did not meet each quality criteria. Give reasons why each success criteria were or were not met.

Quality criteria	Met?	Analysis
1. Generalises well to unseen data	Yes	The model achieved 91.8% ROC AUC, falling short of the 95% target but still demonstrating good generalization to unseen data. This accuracy rate indicates the model learned meaningful patterns rather than overfitting to training data, with minimal performance gap between training and validation sets, making it suitable for real-world application.
2. Interpretability	Yes	The model uses LogisticRegression which provides interpretability through its coefficient weights (Fig. 4), directly indicating each feature's importance and direction of influence on predictions. This makes the model transparent and trustworthy.
3. Continuous improvement	Yes	LogisticRegression supports incremental learning and easy retraining with new data. The computational efficiency allows for frequent model updates (1.4s time to fit), while the linear structure makes it simple to incorporate new features or retrain on expanded datasets. This enables the model to adapt to changing patterns and maintain performance over time.