May 5, 2025

Jaxon Adams

C964: Computer Science Capstone Template

**Warning:** Though it is not stated in the official resources, evaluators do not like outlines.  Write narratively using paragraphs with complete sentences. Use these [C964 examples](https://ashejim.github.io/C964/resources.html#examples) to see what evaluators typically expect.

Task 2 parts A, B, C, and D

[Part A: Letter of Transmittal 1](#_Toc98598250)

[Letter of Transmittal Requirements 2](#_Toc1738085866)

[Letter Template 2](#_Toc1133266111)

[Part B: Project Proposal Plan 3](#_Toc1370766476)

[Project Summary 4](#_Toc904507251)

[Data Summary 4](#_Toc1736393957)

[Implementation 4](#_Toc1241988654)

[Timeline 4](#_Toc1357178365)

[Evaluation Plan 5](#_Toc623361460)

[Resources and Costs 5](#_Toc1538507987)

[Part C: Application 5](#_Toc1471073175)

[Part D: Post-Implementation Report 6](#_Toc651895932)

[Solution Summary 7](#_Toc1134136520)

[Data Summary 7](#_Toc182221765)

[Machine Learning 7](#_Toc1505466430)

[Validation 7](#_Toc391434166)

[Visualizations 7](#_Toc201059345)

[User Guide 7](#_Toc1365484010)

[Reference Page 8](#_Toc1702353417)

# Part A: Letter of Transmittal

## Letter of Transmittal Requirements

The *Letter of Transmittal* should convince senior leadership to approve your project. Write a brief cover letter (suggested length 1-2 pages) describing the problem, how the application (part C) applies to the situation, the practical benefits to the organization, and a brief implementation plan. Include all artifacts typical of a professional (business) letter, e.g., subject line, date, greeting, signature, etc.

The letter should be concise and target a non-technical audience. Include the following:

* A summary of the problem.
* A proposed solution centering around your application.
* How the proposed solution benefits the organization.
* A summary of the costs, timeline, data, and any ethical concerns (if relevant).
* Your relevant expertise.

## Letter Template

[Today’s date]

[Recipient’s name]

[Company name]

[Address]

Dear [Recipient’s name],

(Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer at sodales leo, nec fermentum tellus. Vivamus leo lorem, semper eget erat nec, ultricies vulputate sapien. Integer eget nisi at erat condimentum dictum. Nam sed ornare leo, non pretium nulla. Nam dolor arcu, condimentum et maximus quis, bibendum ut risus. Nam hendrerit ac erat sit amet luctus. Quisque lacinia sapien sed nisl porta, id rutrum odio tristique. Sed sodales nisi a condimentum fermentum. Suspendisse lobortis diam in orci consectetur congue. Sed ligula felis, accumsan eu venenatis ac, hendrerit ac nulla. In ultrices, sem a semper ultrices, orci ex dictum sapien, ac dapibus ligula lectus at est. Quisque at posuere purus, vitae ultricies risus. Donec tincidunt, ipsum eget euismod lobortis, tortor nisl luctus sem, quis blandit tortor dui eu dui.

Praesent sagittis, leo vitae sodales cursus, purus orci rhoncus ligula, quis facilisis ante est nec lectus. Curabitur augue quam, ultricies at arcu eget, molestie eleifend neque. Sed interdum tempor purus, et luctus tellus feugiat imperdiet. Maecenas scelerisque viverra orci tincidunt luctus. Vestibulum sodales eros ut ex luctus tempor. Curabitur eget leo vehicula, malesuada urna ut, eleifend nisi. Quisque sapien tellus, ornare ac magna quis, ultricies consectetur mauris. Nunc erat ligula, mattis id tempor ut, venenatis ac mi. Sed sit amet odio ac ligula tincidunt iaculis sit amet vel ipsum.

Etiam lobortis aliquam metus, eu aliquam ante aliquam ut. Nam tristique sagittis mauris vel tempor. Quisque rhoncus, justo sed lobortis porta, nulla libero pulvinar tortor, ut ullamcorper justo enim ut erat. Praesent lobortis ut leo in aliquet. Suspendisse aliquet velit nulla, a rhoncus nibh vestibulum iaculis. Praesent mollis nibh nec ultrices blandit. Pellentesque felis elit, pretium at risus in, commodo consectetur tortor. Etiam fringilla mi quis erat mollis ultricies. Phasellus vestibulum elementum commodo. Sed congue vulputate orci in porta. Pellentesque scelerisque facilisis justo, a bibendum ligula tempus quis. Aenean efficitur eleifend lorem, et tempus risus consequat quis. Cras varius metus sapien, ac malesuada sem volutpat sed.

Sincerely,

[Sign here: e.g., Jane Smith]

[Your name, title]

# Part B: Project Proposal Plan

## Project Summary

This project will address the problem of effectively assessing credit risk by predicting the likelihood of loan default before funds are issued. As our organization processes a high volume of personal loan applications, we have a need for a data-driven tool to improve decision-making during the loan underwriting process. Our current risk assessment approach is rule-based and lacks the adaptability and predictive accuracy of modern machine learning systems.

To meet our needs, this project will deliver the following:

* A trained machine learning model capable of predicting loan default based on borrower characteristics and loan parameters.
* A Python-based application pipeline that automates data preprocessing, model training, and evaluation.
* Visualizations for performance metrics, feature importance, and threshold tuning.
* A serialized model file that can be reused in production systems or integrated into a larger loan decision engine.
* A simple API implemented with Flask to expose model prediction and provide relevant visualizations.
* A basic frontend web application using HTML, CSS, and vanilla JavaScript to consume the Flask API and provide an intuitive user interface for requesting model predictions.
* A user guide documenting how to operate the application, interpret the outputs, and retrain the model on updated data.

The application will help us improve the accuracy of our loan approval decisions, reduce risk exposure, and optimize return on investment by identifying potentially high-risk applicants before issuing funds. In doing so, this project will modernize and automate part of our credit risk evaluation process.

## Data Summary

The data for this project will come from Lending Club’s publicly available historical loan performance dataset, hosted on Kaggle under the name “Lending Club Loan Data 2007-2020Q3”. This dataset contains over a decade of anonymized loan records, including borrower credit profiles, loan terms, and repayment outcomes.

The raw data will be cleaned and processed through the following stages:

* **Design Phase:** Features relevant to predicting loan default will be selected, and a binary classification target will be defined based on loan status.
* **Development Phase:** Categorical data will be encoded using ordinal encoding, missing values will be imputed using appropriate strategies, and SMOTE will be applied to address class imbalance.
* **Maintenance Phase:** The model will be serializable and reusable, allowing for retraining with updated data. All preprocessing steps will be embedded in a pipeline to ensure consistency.

The dataset meets this project’s needs by offering a wide variety of real-world borrowers and loan features along with labeled outcomes, making it ideal for supervised learning. Common anomalies such as missing data and skewed class distributions will be handled using industry-standard practices. Outliers will be retained unless they represent data corruption, in which case they will be filtered out during preprocessing.

There are no ethical or legal concerns associated with the use of this data, as it is anonymized and publicly available for educational and research purposes under Kaggle’s terms of use.

## Implementation

* Describe an industry-standard methodology to be used.
* An outline of the project’s implementation plan. The focus can be the project’s development or the implementation of the machine learning solution.

## Timeline

* Provide a projected timeline. Include each milestone and deliverable, its dependencies, resources, start and end dates, and duration. (a table is not required but encouraged).
* Dates should be in the future. Write ‘NA’ where an item is not applicable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Milestone or deliverable** | **Project Dependencies** | **Resources** | **Start and End Date** | **Duration** |
|  |  |  |  |  |
|  |  |  |  |  |

## Evaluation Plan

* Describe the verification method(s) to be used at each stage of development.
* Describe the validation method to be used upon completion of the project.

## Costs

Include the itemized costs of the project. Include specific item names where applicable, e.g., ‘PyCharm Professional Ed. 2024.3.5.’

* Itemize hardware and software costs.
* Itemize estimated labor time and costs.
* Itemize estimated environment costs of the application, e.g., deployment, hosting, maintenance, etc.

# Part C: Application

Part C is your submitted application. This part of the document can be left blank or used to include a list of any submitted files or links.

The minimal requirements of the submitted *application* are as follows:

1. **The application functions as described.** Following the ‘User Guide’ in part D, the evaluator must be able to review your application on a Windows 10 machine successfully.
2. **A mathematical algorithm applied to data,** e.g., supervised, unsupervised, or reinforced machine learning method.
3. **A “user interface.”** Following the ‘User Guide’ in part D, the client must be able to use the application to solve the proposed problem (as described in parts A, B, and D). For example, the client can input variables, and the application outputs a prediction.
4. **Three visualizations.** The visualizations can be included separately when including them in the application is not ideal or possible; e.g., the visualizations describe proprietary data, but the application is customer-facing.
5. **Submitted files and links are static and accessible.** All data, source code, and links must be accessible to evaluators on a Windows 10 machine. If parts of the project can be modified after submission, matching source files must be submitted. For example, if the application is a website or hosted notebook, the `.html` or `.ipynb` files must be submitted directly to assessments.

Ideally, submitted applications should be reviewable using either Windows or Mac OS, e.g., Jupyter notebooks, webpages, Python projects, etc. If the source files exceed the 200 MB limit, provide screenshots or a Panopto video of the functioning application and contact your course instructor.

# Part D: Post-Implementation Report

This project addressed the problem of assessing credit risk by predicting the likelihood of loan default based on historical borrower data. Financial institutions need practical tools to evaluate loan applications and manage portfolio risk. Traditional credit scoring methods can be limited in flexibility and predictive power, especially when dealing with large, diverse datasets. This project implemented a supervised machine learning solution using a Random Forest classifier trained on real-world LendingClub loan data to improve accuracy and scalability.

The raw historical loan data was downloaded from Kaggle using this utility script written in Python:

A screen shot of a computer program

AI-generated content may be incorrect.

Initially, the whole dataset was used in model training. After finding complications in deploying the model due to its size, being trained on roughly 1.5 million loans, the entire dataset was instead sampled after being downloaded.

Several steps were identified and implemented to clean and preprocess the dataset throughout the application development lifecycle. First, several columns were dropped to avoid affecting the model through unnecessary noise and data leakage:

A screen shot of a computer screen

AI-generated content may be incorrect.

Next, several data points were cleaned and formatted to make the model more easily understood. Two new features were also engineered and added to the dataset to boost its predictive performance:

A screen shot of a computer program

AI-generated content may be incorrect.

Next, the data was split into training and test data, using a standard 80/20 split. Categorical and numerical features were also identified, allowing missing fields to be correctly imputed based on the data type. Categorical data was also passed through an ordinal encoder to translate the data into a format the model can more easily reason about. SMOTE was also applied in the preprocessing pipeline to address the inherent class imbalance of the problem, where most loans will be paid in full and relatively few will default. This final stage of data preprocessing is defined in the following method:

A screen shot of a computer code

AI-generated content may be incorrect.

Upon completion of these preprocessing steps, the model was trained and tested on the data:

A screen shot of a computer code

AI-generated content may be incorrect.

The trained model outputs a probability of default for each applicant, with a decision threshold that can be configured to suit the client’s risk tolerance best. This threshold-based prediction is beneficial for institutions wanting to customize approval strategies. Visualizations such as precision-recall curves, feature importance plots, and probability distributions are generated at the end of the model training process to support interpretability and operational decision-making. The application provides a modular, reproducible, and effective pipeline for improving loan default risk assessment using machine learning.

## Machine Learning

The machine learning model’s predictive functionality is accessed through the method “predict()” in the file “src/server/app.py”:

A computer screen shot of code

AI-generated content may be incorrect.

This method exposes an endpoint in the Flask API for model prediction. This allows loan application data to be sent in a JSON payload to the server, which queries the model and returns a prediction to the user. An example of the model’s output is shown in the Postman request below:

A screenshot of a computer

AI-generated content may be incorrect.

This method achieves this functionality by using the list of features cached on the server to format the request data into a Pandas data frame. The data frame is submitted to the cached model through the model’s built-in “predict\_proba()” method, where the resulting probability is converted to a Boolean true/false prediction, where “true” means the borrower will likely default on their loan.

This method was selected and developed to make model predictions easily accessible to a frontend web application. Running computationally expensive methods, such as my machine learning model, on the server and making such functionality available to the frontend application through an API is a standard software engineering practice, so a prediction method that creates an API endpoint was a desirable way to expose machine learning functionality in my project.

## Validation

The machine learning model used in this project is a Random Forest classifier, which belongs to the supervised learning category. This model type is well-suited for binary classification tasks like predicting loan defaults because it can capture nonlinear patterns and handle imbalanced data effectively, especially when paired with techniques such as SMOTE (Synthetic Minority Oversampling Technique), which was incorporated into the pipeline to address class imbalance. The model was trained and evaluated using a train-test split, with separate training and test datasets derived from the original LendingClub loan data. This approach provided a precise, unbiased evaluation of the model’s generalization performance on unseen data.

Several evaluation metrics were used to measure model performance. The primary metric was the Area Under the Receiver Operating Characteristic Curve, or AUC-ROC. This metric is appropriate for imbalanced classification problems because it measures the model’s ability to distinguish between classes across different thresholds. Additional metrics, including accuracy, precision, recall, and F1-score, were also calculated to provide a well-rounded performance assessment. The final model achieved an AUC-ROC score of 0.7038, indicating strong discriminative ability. The accuracy was 69%, with a precision of 0.77, a recall of 0.69, and an F1-score of 0.72. These results demonstrate the model’s effectiveness in predicting loan defaults while managing the trade-off between false positives and false negatives, a critical consideration in financial risk modeling.

## Visualizations

Several key visualizations were created to understand better the model’s performance and the data it was trained on. These visualizations can be accessed in a few ways. First, they are available in this project’s “data/visualizations” subdirectory. Second, they can be viewed on the web application dashboard; see the user guide below for information on accessing this project live or in a local environment. Finally, links to the visualizations in a public S3 bucket can be retrieved by sending a request to the “GET /api/visualizations” endpoint in the application’s Flask API. These visualizations are also provided below:

FICO Distribution in Training Dataset:

A graph of a credit score

AI-generated content may be incorrect.

Common Feature Correlation Heatmap:

A screenshot of a computer screen

AI-generated content may be incorrect.

Confusion Matrix:

A blue squares with numbers and a number on them

AI-generated content may be incorrect.

Feature Importance:

A graph of a bar graph

AI-generated content may be incorrect.

Precision-Recall Curve:

A graph with a line

AI-generated content may be incorrect.

Probability Distribution:

A graph of a graph

AI-generated content may be incorrect.

General Performance Metrics:

A chart of performance metrics

AI-generated content may be incorrect.

## User Guide

The application can be used either in a live environment or hosted locally. To use the live application, navigate to the following page in your web browser of choice:

<https://wgu-capstone-client.onrender.com/>

If you wish to run the application locally, first ensure you have a recent version of Python (3.11+) and Node.js (19+) installed on your computer. Python can be downloaded from <https://www.python.org/downloads/>, and Node.js from <https://nodejs.org/en/download/>. Next, perform the following steps:

1. Open a command prompt (PowerShell, Git Bash, etc.) and navigate to the project directory. For example, run the command “cd path/to/project”.
2. Initialize a Python virtual environment with the command “python -m venv .venv”.
3. Activate your new virtual environment; for example, run “./.venv/Scripts/activate”. The actual command may vary depending on your environment. For more details, visit <https://docs.python.org/3/library/venv.html>.
4. Run the following command to install dependencies, download a sample of the full LendingClub dataset, train a Random Forest classifier, and run the full-stack application locally: “npm run for:evaluator”. This process may take several minutes. When the development environment is ready, it should automatically open the web page in your preferred browser.

Once you have the web application (hosted locally or live) open in your web browser, perform the following steps to simulate a client using the application:

1. At the top right of the web page, click the link “Loan Default Prediction” to navigate to the next page.
2. Review the possible inputs for submitting a loan application for prediction. For convenience, defaults are populated for most fields.
3. Click the “Submit Application” button at the top of the form to submit the provided data to the ML model for prediction. After a moment, you should see the model’s prediction listed above the application form.
4. Change a few input fields to simulate a risky borrower; for example, set “Loan Amount” to “40000” and submit the application. Note how the model predicts the loan will default. Now, lower the interest rate to “7.99” and submit the application again; the model should now expect the loan will not default.

# Reference Page

Include references for cited works, e.g., (Author, year), following an accepted writing style. References are unnecessary; this page can be removed if no references are used. To cite sources used for code, you should include the references as code comments within the source code.