March 12, 2025

C964: Computer Science Capstone

Task 2 parts A, B, C and D

Part A: Letter of Transmittal

3/12/2025

Aden Smith

Apex Banking

341 Main St Denver, Colorado 80014

Dear Aden,

Apex Banking strives to provide quality credit card transaction services to clients worldwide. Currently, Apex Banking faces a problem with inadvertently processing fraudulent charges on behalf of the clients for whom we provide services. Correcting these processed fraudulent charges is resource intensive, can lead to legal issues, and detracts us from focusing on other important business areas. These problems can ultimately produce a noticeable decline in the quality we strive for. While it's nearly impossible to prevent processing every fraudulent charge between all clients, there is a solution to mitigate or avoid this costly risk. I'm proposing the creation of an application that predicts the risk of fraud based on a transaction's location, payment method, and industry.

This application will give Apex Banking insight into which types of transactions are more high-risk. Analysis of the likelihood of processing a fraudulent charge allows Apex Banking to make critical decisions, such as which businesses to provide services for. If a potential client comes to Apex Banking seeking a credit card transaction solution, we could run the fraud prediction application by inputting the city where the client is located, their current payment method, and the client's industry. Based on the fraud prediction results for those parameters, Apex Banking can decide to provide services to the client or reject working with the client if the risk of processing a fraudulent charge is too high. Apex Banking could also check if the risk of fraud is lowered when a different payment method is used. If a point of sales (POS) transaction is less risky than an online transaction for that client's location and industry, Apex Banking could stipulate that they'll only process credit card transactions for that client's POS sales. The fraud prediction application will significantly benefit Apex Banking by saving the company time and money through better insights into key decisions.

The overall cost of developing this application is expected to be $107,400. This will cover the salary of three developers, a QA tester, product licenses, and hardware. The estimated timeline is three months, with two

months set aside for development and one month for the final validation testing. The data used to train the Random Forest model, which the program will use to predict the risk of a fraudulent charge, can be trained with a dataset sourced from the web or from transactional data Apex Banking has collected and stored. This data must be cleaned before training the model by removing any irrelevant or missing information. To best protect sensitive information, any dataset obtained will have all personally identifiable information stripped from it. This will help alleviate ethical concerns and ensure this application isn't breaking regulations. All data generated from the application will be stored locally to guarantee that only Apex Banking employees have physical access.

I'd be the best candidate to head this project due to my relevant experience. I have knowledge of the different kinds of machine learning models currently available and which would best suit this project. I'm well-versed in several programming languages like Python, Java, and C++. I have extensive experience with the software development process from start to finish and know the best approach for developing this application. I have also written a design and test plan for previous roles. With my education and experience, I guarantee this application will be designed efficiently while keeping the project on time.

Sincerely,

Zachary Tracy

Zachary Tracy, Junior Developer

Part B: Project Proposal Plan

Apex Banking, a company that processes client credit card transactions, is losing resources and money by inadvertently processing fraudulent charges. Processing these charges leads to investigations that require a person or even a team of people to review the transactions. The legal team might also have to dedicate time to dealing with the fraudulent charges that have been processed. By having a fraud prediction application, Apex Banking can significantly reduce the number of processed fraudulent charges by selectively choosing to work with clients who have a lower risk of credit card fraud. This can be accomplished by using the application to estimate the odds of processing a fraudulent charge based on a client's location, payment method, and industry.

Deliverables for this project include an executable file and a user guide. The executable file will be for Windows operating systems. The executable will contain all the necessary dependencies for the application to run on any computer. Through a command line interface (CLI), the user can either start a web server to interact with the front-end GUI, perform maintenance for the system, or view information and visuals on the machine learning model. The user guide will outline how to start the application and navigate the different options.

This fraud prediction application will benefit Apex Banking by allowing them to gather additional information on the risk a client might pose for processing fraudulent charges. Based on this data, Apex Banking can make better decisions on which clients to provide services for or if a client needs to make changes to how they process payments.

Data Summary

The raw data can be sourced from the internet or from the transactions that Apex Banking handles. If data is used from the internet, a website like Kaggle.com can be utilized to find credit card transactional data as it allows users to upload and share large datasets. The benefit of using a dataset from the internet is that it's quick and easy to obtain. However, it should be reviewed for accuracy. Finding a dataset with all the information needed for the application might also be challenging. Suppose Apex Banking decides to use its own transactional data. In that case, a dataset can be created from prior transactions if that information is saved, or transactional data collection can begin specifically for this project. A column stating if the transaction was fraudulent must be included, and all personally identifiable information must be stripped from the dataset. No matter which way the data is sourced from, there will always be more non-fraudulent charges compared to fraudulent ones. Due to this, the data set will have to be balanced before being used in a machine-learning model.

During the development lifecycle for this application, the data used to train the model will have to be processed and managed. In the design phase, a data structure must be chosen, and the required information must be outlined. A CSV file will hold the dataset for the fraud prediction application. At a minimum, the data needed to train the model are columns on detected fraud, location, payment method, and industry. Other columns can be beneficial to increase the

model's accuracy, such as time of day, risk score, or device type used. The source of the data will also be decided during the design stage. In the development phase, the picked dataset must be cleaned and prepared for use in the machine learning model. First, any personally identifiable information (PII) must be removed to ensure all regulations are met. Next, any missing or irrelevant data will be dropped from the dataset to decrease inaccuracies. Afterward, specific columns must be encoded if they are categorical variables, while columns with numerical variables are to be scaled. This allows the model to make use of columns with string values and, more accurately, use floats and integers. A new CSV file can be generated with these modifications and fed into the machine learning model. For the maintenance phase, any recently acquired transactional data will be added to the dataset and then cleaned for use in the model. Giving the model more data to train on can further increase its accuracy. All information and datasets will be stored locally to keep the data within Apex Banking.

Obtaining the data from a dataset found online meets the project's needs, as fraud detection datasets are already available on websites like Kaggle.com. These datasets are cleaned of personally identifiable information (PII) and can be compared with ‘usability’ scores. Before using a desired dataset from the internet, it will need to be reviewed for completeness, accuracy, and relevancy. Relevancy is determined by how well the data correlates with the issue the machine learning model tries to solve. In this case, datasets with the necessary information are available to help predict credit card fraud. The project's needs will also be met if Apex Banking wants to use data from its own transactions. This data must be formatted into a CSV file, stripped of personally identifiable information (PII), and then any unnecessary information must be removed. Company resources will need to be allocated to collect this data if it’s not already present. Relevancy isn’t a concern as Apex Banking directly deals with credit card transactions for clients, meaning they will have the infrastructure needed to predict credit card fraud. Both methods for obtaining the raw data will not face the challenge of outliers as the target variable is binary. Incomplete data will be removed from the dataset entirely to avoid any potential issues caused by it.

There will be no ethical or legal concerns when collecting this data as all personally identifiable information (PII) will be removed before running the cleaned data through the machine learning model. Any data obtained through online sources will be reviewed thoroughly for PII, and all data coming from Apex Banking will be stripped of PII before use. If Apex Banking currently stores transactional data, its database will have already undergone the correct procedures for storing customer information security to comply with all laws. If transactional data will be collected by Apex Banking solely for this project, then separate hardware needs to be purchased and configured to meet all regulatory requirements.

Implementation

The agile methodology will be used to implement this project. The agile method is a popular industry standard for developing software as it’s well-structured but also allows for excellent flexibility during development. This is possible by breaking down the project into smaller segments called sprints. Changes in functionality and requirements can be easily integrated into the project, even towards the end of development. Also, validation testing can occur for each sprint instead of waiting for the entire project to be finished.

The fraud prediction application will be broken up into four sprints as the project has four major milestones. The first sprint will consist of creating a Python file that reads a raw fraud detection CSV file and then creates and saves the cleaned data to a new CSV file. The second sprint will focus on creating and training a machine-learning model. The model being used is a Random Forest, as it's simple yet effective at classifying information and then making a prediction. In this case, the classification is what variables are or are not associated with fraudulent credit card charges. The third sprint will aim to complete a web-facing front-end and its associated functionalities. This internal website will allow users to easily select their parameters through dropdown menus and then see a prediction alongside three visuals. The fourth sprint will complete the task of tying all the previously completed functionalities together and adding maintenance functionality. This will be done through a CLI interface with a main menu and submenus. The user will be presented with options such as starting up the internal website, viewing graphs and information about the Random Forest model, and re-running the model or CSV cleaning protocol.

Timeline

Milestone or deliverable	Duration (hours or days)	Projected start date	Anticipated end date
Collecting or finding a suitable raw dataset	4 days	04/01/2025	04/04/2025
Sprint #1: Cleaning dataset functionality	10 days	04/07/2025	04/18/2025
Sprint #2: Random Forest creation and training	15 days	04/21/2025	05/09/2025
Sprint #3: Development of a GUI and its functionalities	10 days	05/12/2025	05/23/2025
Sprint #4: CLI menus	10 days	05/26/2025	06/06/2025

and model maintenance functionality			
Validation testing of the final product	15 days	06/09/2025	06/27/2025
Creation of the user guide	5 days	06/30/2025	07/04/2025
Delivering the application and user guide	1 day	07/07/2025	07/07/2025

Evaluation Plan

To verify the application's separate functionalities are working as intended, each sprint needs its produced code validated. For the first sprint, verification includes checking to ensure the created clean CSV file doesn't have any missing data, all included columns are present, and no personally identifiable information (PII) is found. The second sprint will require verification of the Random Forest by checking the model's accuracy and ensuring it is balanced. This can be achieved through built-in functions of the sklearn.metrics library for Python. The machine learning model also must be checked for recall and the f1 score. The third sprint needs verification that no errors occur in the GUI when a prediction is made and that visuals are generated correctly by testing every combination of available options on the dropdown menus. Accuracies for generated predictions will be verified by comparing them to known credit card fraud percentages for several different parameter combinations. For the fourth sprint, verification will occur by testing each menu option to ensure the associated functionality happens. No errors or hangups should be present. The maintenance functionality should display information and graphs accurately and neatly.

Before delivering the application, the complied executable file for Windows must be validated. There should be no discrepancies between the application running on the executable file and in the integrated development environment (IDE). This includes bugs and information accuracy. To validate this, the executable file must have all menu options selected to verify the expected result happens for each one. Afterward, the executable version and IDE version of the application must be run in parallel to confirm the outputted data from the executable matches the data generated in the IDE version. All files and functions are validated and verified at the end of

each sprint. If the executable file displays the same information without bugs, then the application is ready for deployment.

Resources and Costs

Hardware and software costs:

Resource	Description	Cost
Servers	A dedicated server for the development of the application.	$4,000
Operating System	The server requires its own copy of Windows 10 Professional.	$200
Networking	Ethernet cables are required to connect the server to the existing network infrastructure at Apex Banking.	$25
JetBrains PyCharm Professional IDE	An IDE to develop the application in Python. Each of the three developers will need a license at $25 a month. The project will take a total of three months	$225
Employee Laptops	The team assigned to develop and test this project will use laptops already given to them. These will be used to write code and documentation.	$0

Labor time and costs:

Resource	Description	Cost
Developer Labor	Each developer has an annual salary of $100,000. There are a total of three developers, and the project took three months to complete. The estimated labor time for each developer is 400 working hours.	$75,000
QA Tester Labor	The QA Tester has an annual salary of $75,000. There is only one QA tester, and the project took three months to complete. The estimated labor time for the QA tester is 160	$18,750

	hours.	

Environment costs of the application:

Resource	Description	Total Cost
Servers	Two dedicated servers will be solely used to run the application.	$8,000
Operating System	Each Server requires its own copy of Windows 10 Professional.	$400
Networking	Ethernet cables are required to connect the servers to the existing network infrastructure at Apex Banking.	$50
Spare Server Parts and Cables	Extra parts and cabling must be on standby for hardware maintenance.	$750

The estimated total cost of developing, testing, and deploying this application is $107,400.

Part C: Application

List of submitted files:

- Fraud_Predict_App_Win (For Windows)
- App_Source_Files.zip

Part D: Post-implementation Report

Solution Summary

Apex Banking faced a problem with inadvertently processing fraudulent credit card charges for their clients. These fraudulent charges were proving to be costly and resource-intensive; hence, discovering a way to decrease them was a top priority. The solution ended up being the creation of an application that can predict the risk of fraudulent charges through a machine learning model. The application is a stand-alone executable file, ensuring that all dependencies are accounted for while being quick and straightforward to deploy. It has a user-friendly GUI for predicting fraud based on three key parameters and maintenance options to view information regarding the machine learning model (Random Forest).

The application provides a solution for Apex Banking, allowing users to view the percentage and associated graphs for credit card fraud prediction based on the selected location, payment method, and industry type. This has addressed the issue Apex Banking was facing because clients can now be analyzed for the risk of fraudulent charges before Apex Banking begins working with them. To run this prediction, an Apex Banking employee must navigate to an internal website and select the parameters that best fit the potential client when the application runs on the dedicated servers. The employee could even check several payment methods and then work with the client only if they switch their transactions to a less risky payment method. Admins have access to the application CLI menus on the servers, meaning they can update the raw data and then create a new cleaned CSV file for the model to use. They can also retrain the Random Forest model, view statistics on the accuracy of the current model, and see three visualizations regarding the current model. The internal website can also be stopped and started from the GUI if the model is updated. The fraud prediction application has given Apex Banking the tools to reduce the risk of processing fraudulent charges by selectively choosing their clients. It's also easy for the IT team at Apex Banking to manage this application through its effective CLI.

Data Summary

The data for this application was sourced from a CSV file found on Kaggle.com titled “Fraud Detection Transactions Dataset”. This dataset was ultimately chosen over Apex Banking’s own transactional data as it was quick to obtain, organized, and already mostly cleaned. This allowed more time to focus on the development of the app rather than collecting transactional data. The online dataset was also free of personally identifiable information (PII) as there is only an anonymous ‘User ID’ column to identify people. This raw data meets the needs for training the machine learning model because it included important columns such as ‘Fraud Label’, ‘Location’, ‘Transaction Type’ and ‘Merchant Category’. Other notable columns were ‘Timestamp' and ‘Risk Score’.

The design phase included deciding on a way to source the raw data by weighing the pros and cons of each collection type and then ensuring the dataset was suitable for training the model. As discussed, a CSV file selected from Kaggle.com was chosen as the best source of the needed

data. For the development phase, the chosen raw dataset was cleaned by removing any rows with empty cells and deleting irrelevant columns such as ‘Transaction ID’. Additional columns were added to increase accuracy by combining or modifying information from different rows. Afterward, numerical data was scaled, and categorical data was encoded to allow the information to be more accurately used for training the model. It was discovered during the training of the model that the dataset was significantly imbalanced due to the larger number of non-fraudulent charges compared to fraudulent ones. To fix this, the number of fraudulent charges was artificially inflated through oversampling. This change helped drastically improve the accuracy of the Random Forest model. There has been no need to process or manage the raw data further in the maintenance phase. The raw dataset and all created data are stored within the temporary folder generated by the application when it is executed. Upon exiting the application, all data is deleted. Suppose an IT admin from Apex Banking wishes to store any data permanently. In that case, they will need to navigate to the temporary folder for the application and copy the file before closing the application. For this reason, the CLI for the application shares the file path to where the files are stored.

Machine Learning

The machine learning model used for this application is the Random Forest model. A Random Forest is considered a supervised learning model, requiring labeled training data to learn patterns. From these learned patterns, the model can make predictions. This project is trying to predict the chances of fraudulent credit card charges using the cleaned “Fraud Detection Transactions Dataset” found on Kaggle.com. The Random Forest model is a subcategory of ensemble learning as it uses multiple decision trees to increase its predictive accuracy.

The sklearn library was used to implement the Random Forest model into this application. This library has functions that make generating a Random Forest simple. After generating the model with the cleaned dataset, it is saved as a PKL file type and added to a temporary folder. The saved model is then loaded to make the predictions in the user GUI based on the chosen parameters. This way, the model doesn’t have to be regenerated each time the user changes the parameters for another prediction. The model is also loaded for some maintenance options, such as viewing the accuracy of the model or seeing the three generated graphs associated with it.

The Random Forest model was chosen for this application as it excels at prediction through classification compared to other machine learning models. This model offers the simplicity of a decision tree but with a higher degree of accuracy as it’s more consistent with capturing actual patterns instead of noise. A Random Forest model also handles imbalanced data well, which is essential considering there will always be many more regular transactions than fraudulent ones in a dataset.

Validation

Validation of the Random Forest model was achieved by using the train-test splitting technique through the sklearn library for Python. After loading the cleaned dataset, it was split between use in model training and testing. To reduce the impact of removing useful data from

model training, only 20% of the dataset was allocated for testing. After splitting the dataset and training the model, accuracy was validated using the accuracy_score() function from the sklearn library. A classification report was also generated using classification_report() to view the recall and f1 score of the model.

The Random Forest model achieved an overall accuracy of 81.7% through trial and error by adjusting different metrics, such as adding additional columns, removing columns with a low correlation score, and testing various decision thresholds. For predicting fraudulent charges, the model has a recall of 68% and a f1 score of 79%. For predicting non-fraudulent charges, a recall of 96% was given with a f1 score of 84%. Research and the development of a different machine-learning model might be required to obtain a more accurate prediction in the future.

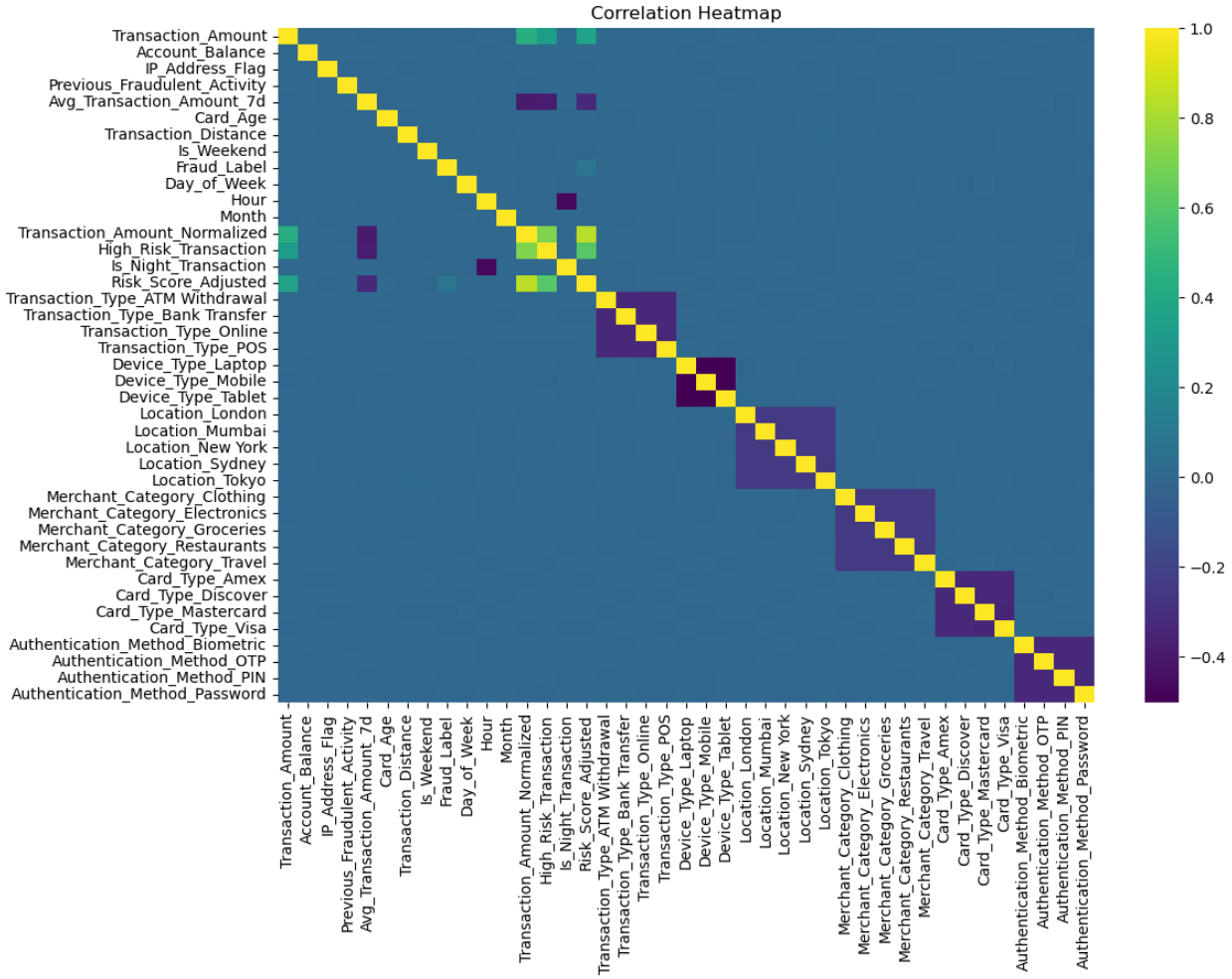
Visualizations

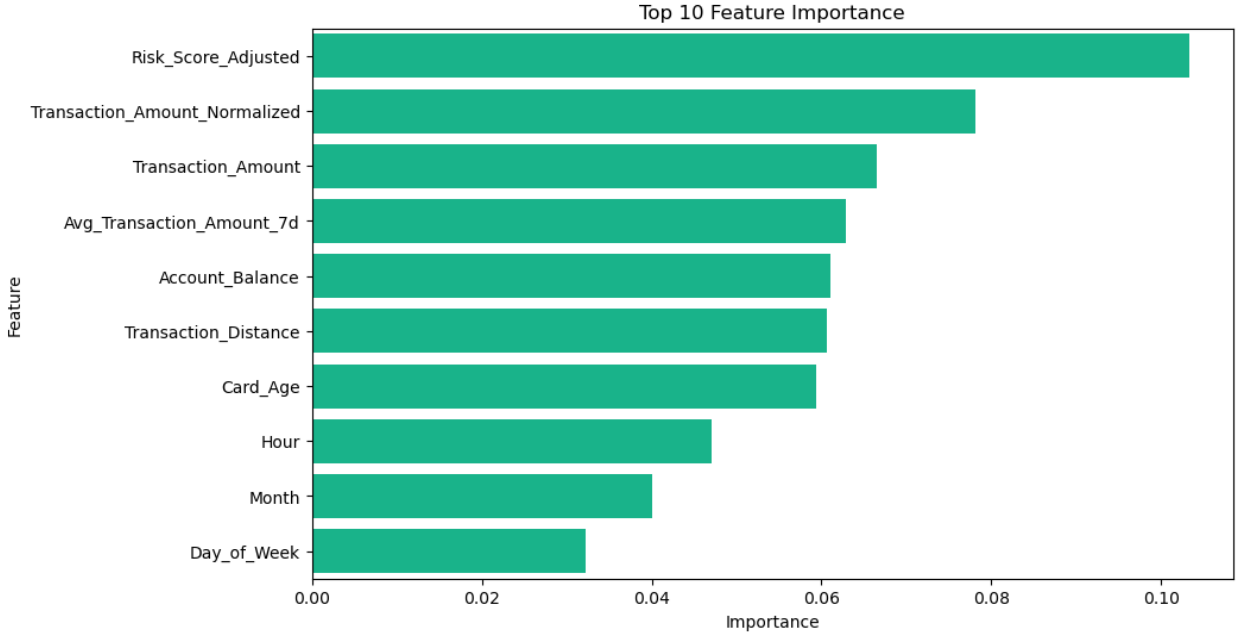
Six unique visualizations can be found in the fraud prediction application. Three graphs can be viewed on the user GUI interface to predict fraud based on selected parameters. These graphs give a filtered visualization of information depending on the combination of parameters selected.

The other three graphs can be viewed in the application's CLI by selecting the following options starting from the “CLI for fraud prediction system” menu (do not include the quotes surrounding each number):

1. Type “2” and hit enter on your keyboard to navigate to the “Maintenance Menu”.
2. Type “2” and hit enter on your keyboard to navigate to the “Graph Menu”.
3. Type either “1”, “2”, or “3” and hit enter on your keyboard to select, generate, then view a graph.

These graphs share information related to the entire generated Random Forest model the application uses.


Correlation Heatmap
Transaction_Amount
Account_Balance
IP_Address_Flag
Previous_Fraudulent_Activity
Avg_Transaction_Amount_7d
Card_Age
Transaction_Distance
Is_Weekend
Fraud_Label
Day_of_Week
Hour
Month
Transaction_Amount_Normalized
High_Risk_Transaction
Is_Night_Transaction
Risk_Score_Adjusted
Transaction_Type_ATM Withdrawal
Transaction_Type_Bank Transfer
Transaction_Type_Online
Transaction_Type_POS
Device_Type_Laptop
Device_Type_Mobile
Device_Type_Tablet
Location_London
Location_Mumbai
Location_New York
Location_Sydney
Location_Tokyo
Merchant_Category_Clothing
Merchant_Category_Electronics
Merchant_Category_Groceries
Merchant_Category_Restaurants
Merchant_Category_Travel
Card_Type_Amex
Card_Type_Discover
Card_Type_Mastercard
Card_Type_Visa
Authentication_Method_Biometric
Authentication_Method_OTP
Authentication_Method_PIN
Authentication_Method_Password
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


Top 10 Feature Importance
Risk_Score_Adjusted
Transaction_Amount_Normalized
Transaction_Amount
Avg_Transaction_Amount_7d
Account_Balance
Transaction_Distance
Card_Age
Hour
Month
Day_of_Week
Feature
0.00
0.02
0.04
0.06
0.08
0.10
Importance

User Guide

To download, start, and stop the application:

Step Number	Action	Expected Result
1	Locate the downloadable executable files for the application in its designated location. Note: This location will be in the current C964 Task 2 attempt for evaluators.	Two downloadable files labeled Fraud_Predict_App_Win and Fraud_Predict_App_Mac will be visible in the designated location.
2	Find and download the correct executable file for Windows named: Fraud_Predict_App_Win.	The executable for Windows will be found and then downloaded to their system.
3	Double-click on the downloaded executable file to start the application. Note: if a warning appears stating “Windows protected your PC” find and click the “Run anyways” option. “More info” may have to be clicked to get “Run anyways” to appear. No publisher information was submitted to Microsoft for this project.	The user’s command line interface will appear, and the program will start.
4	Please wait 30-60 seconds for the “CLI for fraud prediction system” menu to appear.	The application will automatically clean the raw dataset and train the model upon

	Note: This is dependent on your system's performance. The timeframe can be anywhere between 15 seconds to 90 seconds. The average is approximately 40 seconds.	startup. After completion, the "CLI for fraud prediction system" menu will appear.
5	To stop the application, type "0" (do not include the quotes) and hit enter on your keyboard while viewing the "CLI for fraud prediction system" menu.	The application will terminate for the user.

To start, use, and shut down the GUI:

Step Number	Action	Expected Result
1	With the application started and the "CLI for fraud prediction system" menu displayed, type "1" (do not include the quotes) and hit enter on your keyboard to start the user GUI interface for predicting fraud based on selected parameters.	The local host server will start up, and a tab on the user's default browser should open if the user's system permits it.
2	A new tab on your browser should now be open with the GUI interface titled "Credit Card Fraud Risk Predictor". Note: If this has not happened, please navigate to your preferred browser and type in the address displayed in the CLI.	The GUI will appear for the user with the title "Credit Card Fraud Risk Predictor". If the user's system doesn't allow the application to launch and/or open a tab in the default browser, the GUI is still accessible by typing in the address shown in the CLI.
3	Select your desired parameters from each dropdown menu to interact with the GUI, then click the "Predict Fraud Risk" button.	When the user selects "Predict Fraud Risk", a fraud prediction percentage will appear on screen and the three graphs will populate.
4	To shut down the GUI, navigate back to the CLI where the application is currently running. Click "Control + C" on your keyboard.	After returning to the CLI and clicking CTRL + C, the local server will shut down, and the user will be returned to the "CLI for fraud prediction system" menu.

To view information about the ML model and re-run functionalities:

Step Number	Action	Expected Result
1	With the application started and the "CLI for fraud prediction system" menu displayed, type "2" (do not include the quotes) and hit enter on your keyboard to navigate to the "Maintenance Menu".	The "Maintenance Menu" will appear in the CLI.
2	Within the "Maintenance Menu", to view text-based information regarding the currently	Information such as overall accuracy, recall, f1 scores, and a correlation matrix

	generated Random Forest model, type “1” (do not include the quotes) and hit enter on your keyboard.	will appear in the CLI for the user. The “Maintenance Menu” will reappear at the end to allow the user to select another option.
3	Within the “Maintenance Menu”, to generate visuals regarding the currently generated Random Forest model, type “2” (do not include the quotes) and hit enter on your keyboard.	A “Graph Menu” will appear for the user to select which graph they would like to generate and view.
4	Within the “Graph Menu”, type either “1”, “2”, or “3” (do not include the quotes) then hit enter on your keyboard to generate and view your desired graph. Note: If your default image viewer doesn’t automatically launch with an image of the generated graph, please navigate to the temporary folder in which the graph was saved to view it. This temporary folder will be displayed on the CLI.	Upon choosing a number to type and then hitting enter, the application will generate the desired graph, save it to a temporary folder in the user's system, and then automatically open an image of the graph in the user's default image viewer. This image can still be viewed in the temporary folder displayed on the CLI if the user’s system denies the application access to launch the image viewer. The “Graph Menu” will reappear after the image is generated to allow the user to choose another option.
5	To exit the “Graph Menu” and return to the “Maintenance Menu”, type “0” (do not include the quotes) and hit enter on your keyboard.	The “Maintenance Menu” will appear on the CLI.
6	Within the “Maintenance Menu”, to re-run the functionality that cleans the raw dataset, type "3" (do not include the quotes) and hit enter on your keyboard.	The raw dataset file will be cleaned again and replaced with the previously saved cleaned dataset CSV file. The “Maintenance Menu” will reappear on the CLI.
7	Within the “Maintenance Menu”, to re-run the functionality that trains the Random Forest model, type "4" (do not include the quotes) and hit enter on your keyboard. Note: retraining the model can take a few moments.	The Random Forest model will be retrained and replaced with the currently saved model. The “Maintenance Menu” will reappear on the CLI.
8	To return to the “CLI for fraud prediction system” menu, type “0” (do not include the quotes) and hit enter on your keyboard.	The “CLI for fraud prediction system” menu will appear on the CLI.

Reference Page

No references were used for this task.