

Credit Card Fraud Predictive Modeling and Deployment

**Interim Report**

Higher Diploma in Science in Data Analytics

Word Count: 7500

Author: Ciaran Finnegan / 10524150

E-mail: [10524150@mydbs.ie](mailto:10524150@mydbs.ie) / [ciaran@feefinnegan.com](mailto:ciaran@feefinnegan.com)

Supervisor: Dr Shahram Azizi Sazi

Report Submission Date: 14th August 2020

**Acknowledgments**

I want to acknowledge the advice and support provided by two fellow work colleagues from Norkom Technologies, Eddie Baggott and Dan Branley. Both worked in the area of software development for Fraud prevention and were able to provide me with the demo data that I was able to re-purpose as a dataset for this data analytics project.

I also wish to acknowledge direction given by my project supervisor Dr Shahram Azizi Sazi, which helped guide my approach to this Interim Report and the with the wider project in general.

# Contents

[Contents 2](#_Toc48316992)

[1. Introduction 4](#_Toc48316993)

[1.1. The Project Deliverable 4](#_Toc48316994)

[1.2. The Aims of the Project 4](#_Toc48316995)

[1.3. Scope of the Project 5](#_Toc48316996)

[1.4. The Project Approach 6](#_Toc48316997)

[1.5. Project Assumptions 7](#_Toc48316998)

[2. Background 8](#_Toc48316999)

[2.1. What Type of ‘Real-World’ Application Does this Project Emulate? 8](#_Toc48317000)

[2.2. Credit Card Fraud Detection: Other Research on Predictive Models 9](#_Toc48317001)

[2.3. Toolkits: Why They Were Chosen 11](#_Toc48317002)

[3. Requirements: Specification and Design 14](#_Toc48317003)

[3.1. Project Mission Statement 14](#_Toc48317004)

[3.2. Project Requirements 15](#_Toc48317005)

[3.3. Project Architecture Diagram 16](#_Toc48317006)

[3.4. Project Design 17](#_Toc48317007)

[3.3.1. User Story Structure 18](#_Toc48317012)

[3.3.2. User Story Roles 19](#_Toc48317013)

[3.3.3. User Story 4 – Initial Data Modelling and User Interface ‘Shell’. 20](#_Toc48317014)

[3.3.4. User Story 5 – Shiny App Prototype 22](#_Toc48317015)

[3.3.5. User Story 6 – Hosted Prototype 24](#_Toc48317016)

[3.3.6. User Story 7 – Feature Enhancements 1 26](#_Toc48317017)

[3.3.7. User Story 8 – Feature Enhancements 2 28](#_Toc48317018)

[3.3.8. User Story 9 – Final Project Refinements 30](#_Toc48317019)

[4. Project Testing and Evaluation 31](#_Toc48317020)

[4.1. Evaluation of User Stories 31](#_Toc48317021)

[4.2. Project Testing 32](#_Toc48317022)

[5. Demonstration of Progress 33](#_Toc48317023)

[5.1. Credit Card Fraud Application: Prototype Location 33](#_Toc48317024)

[5.2. Credit Card Fraud Application: User Guide (Prototype) 33](#_Toc48317025)

[5.3. Project Plan 2020: Current Status – 14th August 2020 34](#_Toc48317026)

[6. Project Re-Design Considerations 35](#_Toc48317027)

[6.1. Initial Proposal 35](#_Toc48317028)

[6.2. Possible Future Design/Deployment Considerations 35](#_Toc48317029)

[7. Appendices 36](#_Toc48317030)

[7.1. Azure Generated Code Segments 36](#_Toc48317031)

[7.2. Shiny R Application Code Files 38](#_Toc48317032)

[7.3. Azure Machine Learning Classic Studio Experiments 39](#_Toc48317033)

[7.4. Credit Card Fraud Datasets 41](#_Toc48317034)

[8. References / Bibliography - Interim Report 42](#_Toc48317035)

# Introduction

## The Project Deliverable

The artefact at the end of this project is an application that invokes a bespoke predictive model and provides a user with an online interface to retrieve a score for whether a given credit card transaction is likely to be fraudulent.

A picture containing computer

Description automatically generated

## The Aims of the Project

The project intends to apply robust Machine Learning techniques to build a predictive model for credit card fraud, using an industry sourced dataset, and deploy this model through an Azure hosted API accessed through a separate bespoke ShinyIO hosted R Application.

Using techniques and knowledge acquired during my DBS Data Science Higher Diploma course I will build as accurate a model as possible.

Extrapolating from programming and data warehousing course material, I will build an R Shiny dashboard as a deployment/interface platform.

The work in this project is intended to demonstrate an application of many of the key tenants of data analytics to which I was exposed during my DBS course work in 2019/2020.

To reflect the growing focus on cloud-based techniques and infrastructure for financial services products, including data analytics applications, this project also aims to work in these entirely online domains;

* RStudio Cloud
* Azure Machine Learning Classic Studio
* ShinyIO

## Scope of the Project

The user interface will be built as a hosted Shiny R dashboard application, which provides two primary functions;

* A means to select a given ‘new’ credit card transaction and assess in real time if this record is likely to be fraudulent.
* Provide a visual analysis of the credit card dataset used to build the predictive card model.

***(The project prototype is currently hosted on*** *Shinyapps.io****. See Section 5 of this document for URL and User Guide).***

The predictive model itself will be built in an *Azure Machine Learning Classic Studio Workspace*. A documented Machine Learning process will be applied to manipulate the source dataset of credit card transactions and build as optimal a predictive model as possible.

Azure workspaces will host the credit card dataset, along with supplementary subsets of the data.

Azure will host a REST Endpoint, which will allow an API call to be written in R and invoked from the R Shiny dashboard application. This API will pass attributes of a ‘new’ credit card transaction and return a score reflecting the likelihood of fraud being present.

The visualisations that will be presented on the Shiny R dashboard are intended to provide a supplementary understanding of the structure of the credit card transaction data.

The Azure Machine Learning Classic Studio provides several in-built visualisation techniques, which I expect to use during the modelling process. However, to improve the richness of my project demonstrations, and to highlight my understanding of R libraries and useful graphical techniques, I will be adding on-screen data visualisation to the R Shiny Dashboard.

These Shiny R dashboard visualisations will improve any explanations I wish to give about the credit card dataset structure during demonstrations.

## The Project Approach

*Data*

The source dataset for modelling has been retrieved from test/demo data used for a discontinued product line in my company (Norkom Technologies).

This product was a rules engine to detect credit card fraud based on the sequential application of business rules. The transaction dataset is extensive both in terms of numbers of columns/attributes and rows and is an ideal repository upon which to build a Machine Learning predictive model.

*Modelling and Deployment*

I wanted to generate a project what was both built and deployed using cloud-based tools and techniques.

The application of Machine Learning workflows to build the predictive fraud model is being carried out in an Azure Machine Learning Classic Studio workspace. (Microsoft Tutorial, 2020)1

The user interface is a Shiny R Dashboard, built in RStudio Cloud and deployed as a ShinyIO application using open source Semantic libraries as described by Krzemiński, 20182.

*Quick Iterative Design/Development*

Arguably the most important project approach to highlight is the intention to deliver working software as early as possible, and at regular intervals thereafter. I have followed my own project experience and common industry practice, as described by Parsons3 in 2019in his book *When To Use Waterfall Vs. Agile*, by adopting a pseudo-AGILE approach to design, development, and deployment.

I feel one of the major risks for the project is that I am building the predictive model with one toolkit (Azure Machine Learning Studio), and deploying an API to be invoked by a completely separate application, an R Shiny dashboard application. This is a deployment model with which I have never worked previously, and inter-connectivity challenges could a major problem.

The demo I have built for this Interim report shows a relatively simplistic approach to building the predictive model. I am deliberately focusing initially on the verification exercise that the project application will work ‘end-to-end’, and that data will flow seamlessly between Azure and my Shiny R dashboard application.

My project plan has been constructed to focus on the ‘Production’ deployment framework first and then enhance the model afterwards. This can be seen in the project plan in Section 5.2 of this document, and expanded on in more detail in Section 3.3.6 (User Story design description).

## Project Assumptions

*Primary Dataset*

The source dataset I am using is also essentially demo data and is thus relatively clean of empty or corrupt values.

Critically, it contains a label column (marked ‘Fraud’) so that each row can be identified as being a historically fraudulent record, or not.

However, as it was intended to interface with several legacy systems the dataset contains many columns that may be redundant. Feature Selection is expected to be a critical exercise to ensure the most effective application of the correct modelling algorithm is a manageable process.

For the initial stages of the project I have carried out manual feature selection based on internal company guidance from a colleague with prior experience in credit card fraud product management.

*Data Sub-Sets*

I often used inverted commas in this report when describing the ‘new’ data that the user selects in the UI before invoking the API to run the predictive fraud model. The reason for this convention is that I have assumed that project development will require the following split of the original credit card fraud dataset;

* 5 rows have been extracted from the original dataset to be stored in Azure and used as the initial ‘pre-loaded’ data during early UI development. This data will be scored using the API call to the predictive fraud model but will not have been used in the modelling process itself.
* 200 rows have also been extracted from the original dataset to act as ‘new’ rows. This data will be split into multiple EXCEL files and will simulate ‘new’ data. The user will select files through the UI and obtain a predictive fraud score. Although this data does actually contain a ‘Fraud’ column it will not be passed to the model but can obviously be used for comparison to assess ‘correctness’ of the model score. Again, this data will not have been used in the modelling process itself.
* Initial modelling will take place on 2.5K rows of the remaining master dataset. *PowerBI* data manipulation has been used to ensure the Fraud/Non-Fraud proportions in the master dataset are maintained. This reduced dataset is used to limit the initial complexity of the first data modelling iteration.
* The entire remaining 100K+ rows of the credit card fraud dataset will be used to generate the final predictive model.

# Background

## What Type of ‘Real-World’ Application Does this Project Emulate?

*Why Credit Card Fraud?*

In his 2019 article4, Keith Stanton stated that Europe saw 17 cards with fraud on every 1,000 cards issued in 2013, and by 2016 this had increased to 47 cards per 1,000 cards, an increase of 176%.

Credit card fraud is a crime that has touched many of us, so I wanted to look at data analytics and how it might be applied in this area.

*Personal Background*

I work within a company that specialises in the software development of financial crime prevention software. To date, much of our fraud prevention products have used rules-based processes to generate alerts for possible fraudulent transactions, but that will gradually change with the emergence of new data analytics product objectives.

This project artefact is a simplified version of what a basic credit card fraud application might look like in a Financial Services domain. A Fraud Investigator can select a given (single) credit card transaction and retrieve a score to reflect if fraud is probably present.

*Real-time Credit Card Fraud Detection – Limitation in my Approach*

The project artefact is essentially an academic exercise in build a predictive credit card fraud model and provide a basic (but attractive) user interface.

In the ‘real world’ daily credit transaction volumes are enormous. According to the 2019 article5 by J. Cherowbrier, in the UK alone nearly 40 million transaction per day were recorded on average in 2016. Many organisations will have a significant base of employees devoted to fraud investigation but still cannot remotely hope to manually inspect each company credit card transaction. In practice, any credit card fraud application will process high volumes of data in *automated batch transactions* and filter out ‘high-alerts’ for human inspection.

My application provides a ‘one-at-time’ process to inspect individual transactions for fraud. This is obviously unrealistic and is more akin to *Customer-Due-Diligence* systems6 (ICA, n.d.) that allow for profiling of new customers where the volume of data is very much lower.

However, I believe that the exercise in building this project artefact is still an extremely useful learning process for me. It is certainly conceivable that ad-hoc fraud detection could be useful in some business use cases.

## Credit Card Fraud Detection: Other Research on Predictive Models

As of mid-August 2020, much of my project work to date has been on building the ‘production framework’, which can be seen in the prototype demo.

However, I also conducted some initial research on comparable work carried out, and publicly available for review, in the area of predictive data analytics models for credit card fraud detection.

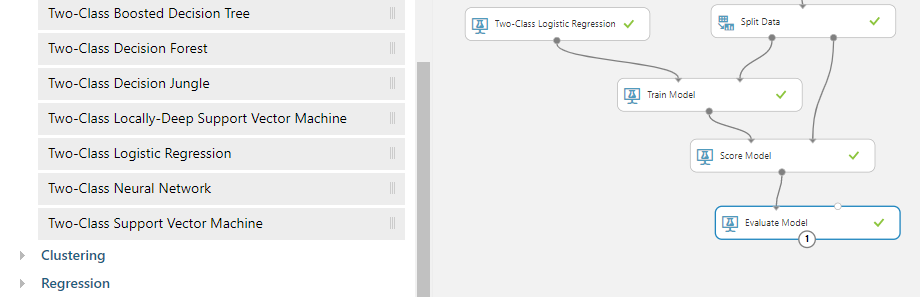
The most accessible research is on Kaggle but that in turn acknowledges the work of the Machine Learning Group (<http://mlg.ulb.ac.be>) of ULB (Université Libre de Bruxelles) in the area of fraud detection.

This group has generated a credit card fraud dataset7 (Machine Learning Group - ULB, 2018) that features in a few Kaggle projects, most notably by Preda8, 2018 and Charminda D9, 2020.

When looking at this analysis, and comparing the structure of my own credit card dataset, I have made the following observations that will influence my approach to generating the model in this project;

* The ULB data is based on an extract of European credit card transaction in 2013. My dataset is based on US credit card transactions in a similar timeframe. Data volumes are relatively comparable.
* The use of ‘chip-and-pin’ in Europe at that time would have been commonplace, but not so in the US. However, the European data only contains numerical values that are the result of a PCA transformation. This was done for reasons of confidentiality, but it means that there will be relatively opportunity to look at cross-Atlantic differences in model accuracy due to certain, easily understandable, attributes.
* The only features in the European dataset that have not been transformed with PCA are ‘Time’ and ‘Amount’ so most of the available data visualisations relate to only that data. However, that should still provide some useful guidelines for my own project visualisations.
* The principle components obtained on the European dataset are numbered V1 to V28, so the number of columns in the dataset is relatively small. The US credit card dataset is essentially all the elements of the transactions and includes a lot of superfluous product interface data. Hence, the number of columns is 380. Analysis on the ULB data did not have to particularly focus on removing highly correlated data and feature importance/selection was more straightforward, which will not be the case with the US dataset.
* The European dataset is highly imbalanced and that has had a major impact on the approach to building the model in the Kaggle projects.
* Gabriel Preda invests considerable time in data balancing techniques before applying a range of predictive algorithms. The imbalance is deemed to make Confusion Matrices less useful so the Area Under the Curve (AUC) metric is preferred. Algorithm selection starts with *RandomForest*, then moves onto *AdaBootClassifier*, and onto other choices before settling on a *LightGBM* model with an AUC score for the test prediction of 0.93.
* Charminda D prefers to look at algorithms specially meant for solving anomaly detection such as *Isolated Forest*, *Elliptic Envelope*, and *One-Class SVM*.
* My US dataset is much less imbalanced. Approximately 15% of the records represent ‘Fraud’, as opposed to less than 1% for the European data. This leads me to think that I will have less of an issue balancing the data and can look more closely at the algorithm selection from Gabriel Preda in the later stages of his project.
* When I built the predictive model for the prototype in Azure Machine Learning Studio I used the supplied ‘Two-Class Logistic Regression’ model, based on the introductory Microsoft tutorials. Feature selection was a manual process, based on business domain knowledge, and was limited to numerical data. For the later stages of the project I intend to greatly develop the machine learning process to build my credit card predictive model. The Kaggle research mentioned above will be a key jumping off point but I will also be referring to useful project entries in the relevant ResearchGate blog (<https://www.researchgate.net/project/Fraud-detection-5>).

*Figure: Azure Machine Learning Studio Algorithms (sample)*



## Toolkits: Why They Were Chosen

*Data Modelling*

The Data Warehousing module in my DBS course was a particularly stimulating subject and the introduction to the CRISP-DM workflows is a topic I have seen replicated in many other external documents/articles.

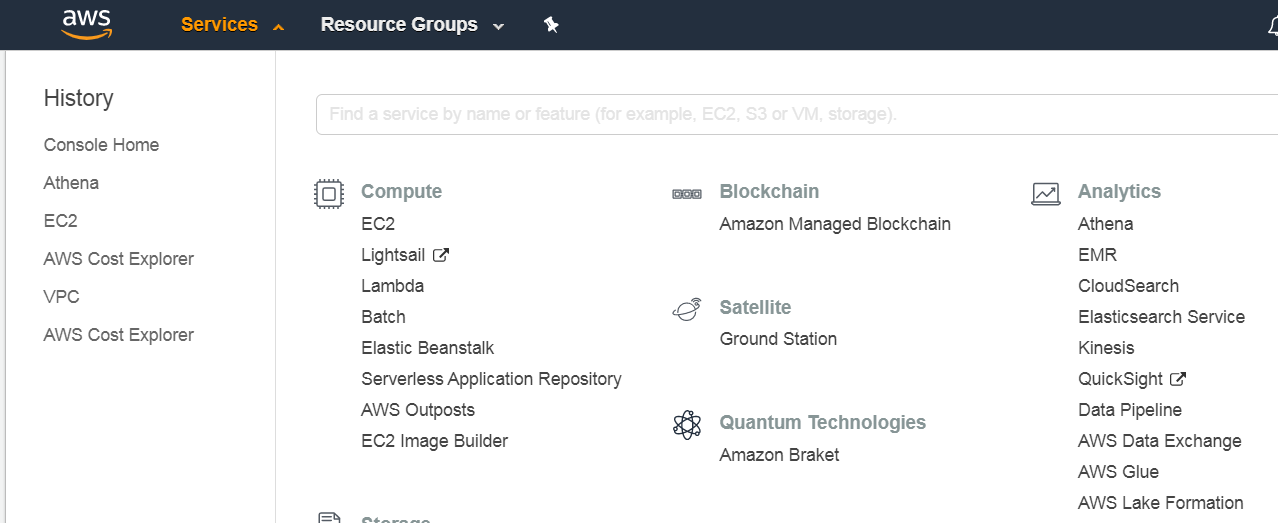
For data modelling, I wanted to use a tool that was somewhat analogous to RapidMiner but arguably had a higher industry profile and was also not limited to a deployment on a single local machine.

The early phases of this project involved a brief cycle of research to look at options for cloud-based environments within which to build my predictive credit card fraud model. The full Machine Learning process would have to be supported, from data ingestion/manipulation and modelling, through to deployment frameworks.

Modelling toolkit options I considered were;

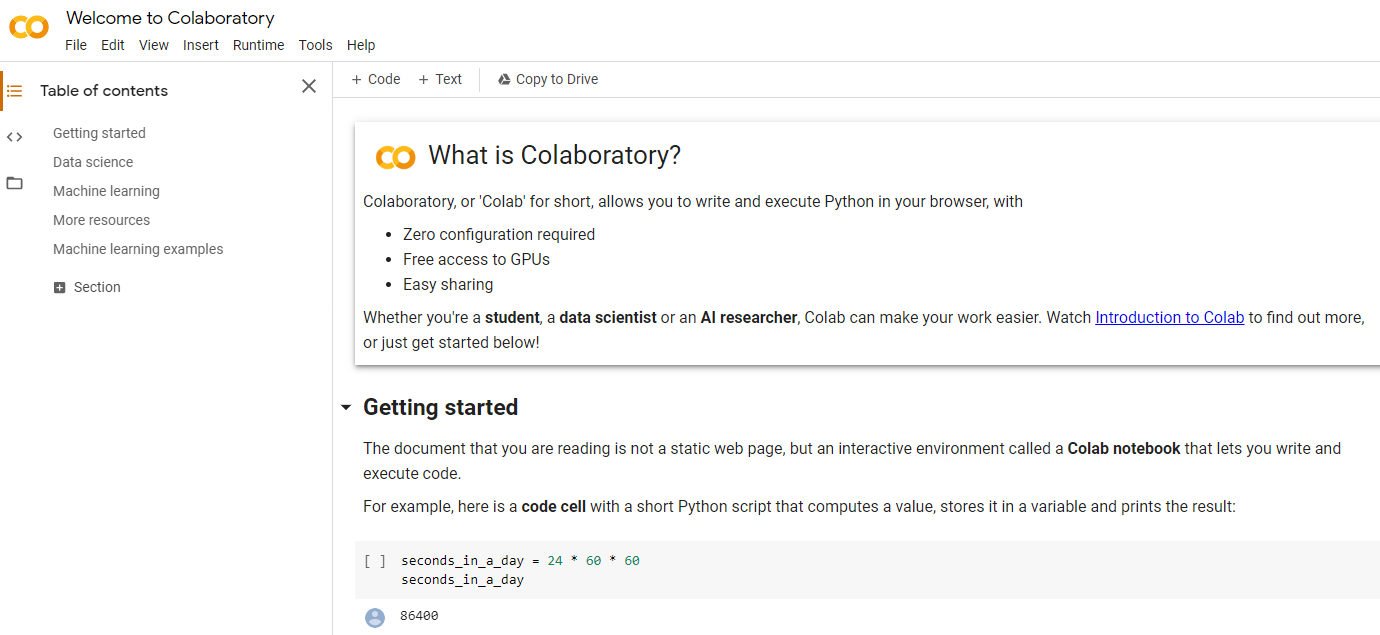
* **AWS** : Analytics Services. (Escapa, 2018)10 Many of my company development environments are set up in our corporate AWS account and I have an active account with access to a range of services. However, I found it relatively difficult to find useful ‘get-started-quickly’ documentation for AWS Analytics.

*Figure: AWS Console*



* Google Colaboratory. (Sharma, 2020)11 This service is easy to access from a browser and free to use for an individual. I ran some initial notebook experiments and briefly considered a Tensor Flow approach to writing a Neural Net solution for credit card fraud. This looked initially promising, but the learning curve became steep quite quickly and l felt that this was a risky option. (A possible M.Sc. project candidate!)

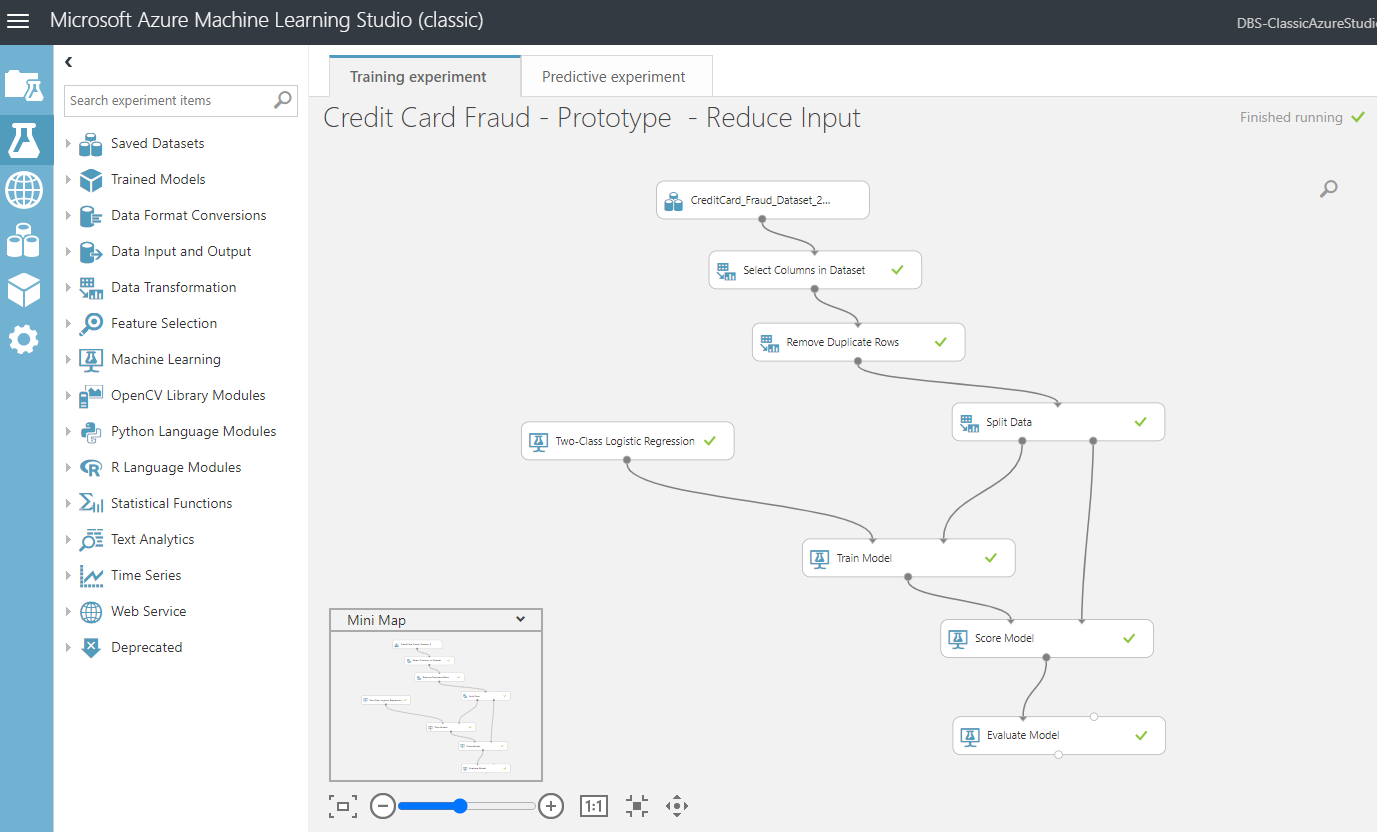
*Figure: Google Colab*



<https://colab.research.google.com/>

* **Azure Machine Learning Classic Studio**: (Microsoft Tutorial, 2020)12 Microsoft Azure is another cloud-based platform on which my company hosts development environments. I work in this environment on a moderately frequent basis and it is the likely cloud platform of choice for future company rollouts. There is a wealth of online material to help any student get started so this platform became my choice for this project environment.

*Figure: Azure Classis Machine Learning Studio*



*User Interface: Programming Languages Environments*

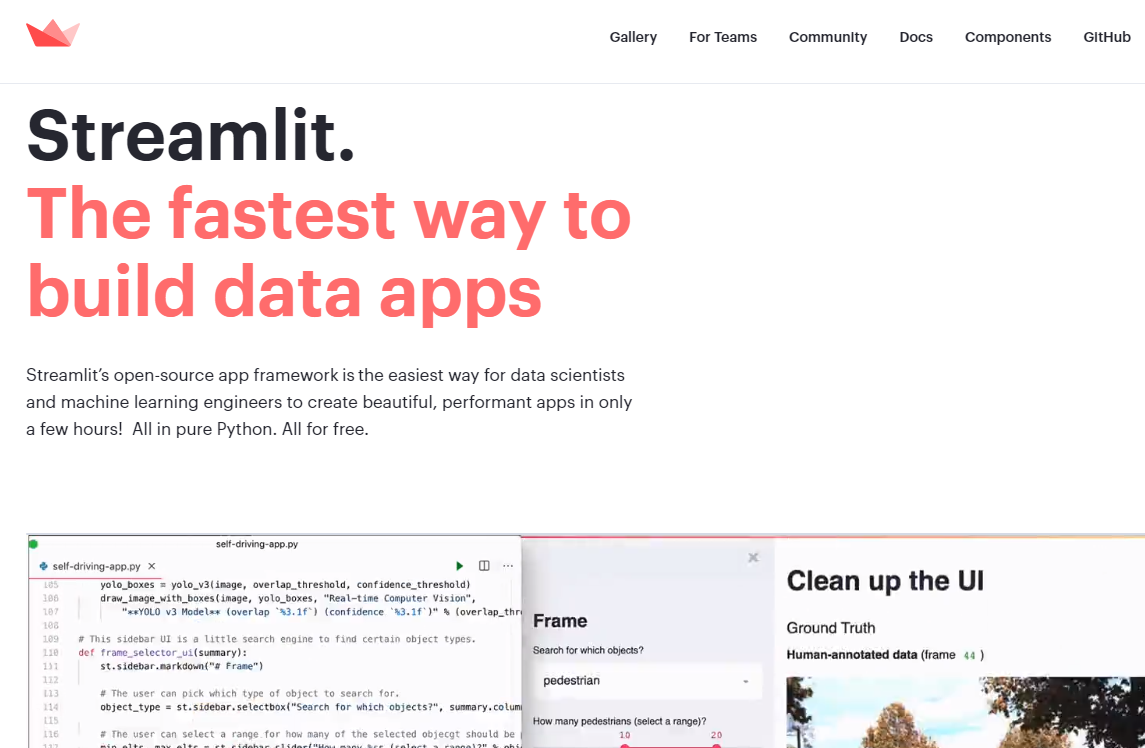
My key criteria for a programming language was that I wanted to be able to deploy a web application, but without undue complexity. Given DBS course content I would obviously be focusing on either Python or R paradigms.

Django and Flask were ruled out almost immediately as I felt this was too great a risk for the project.

Primary considerations were;

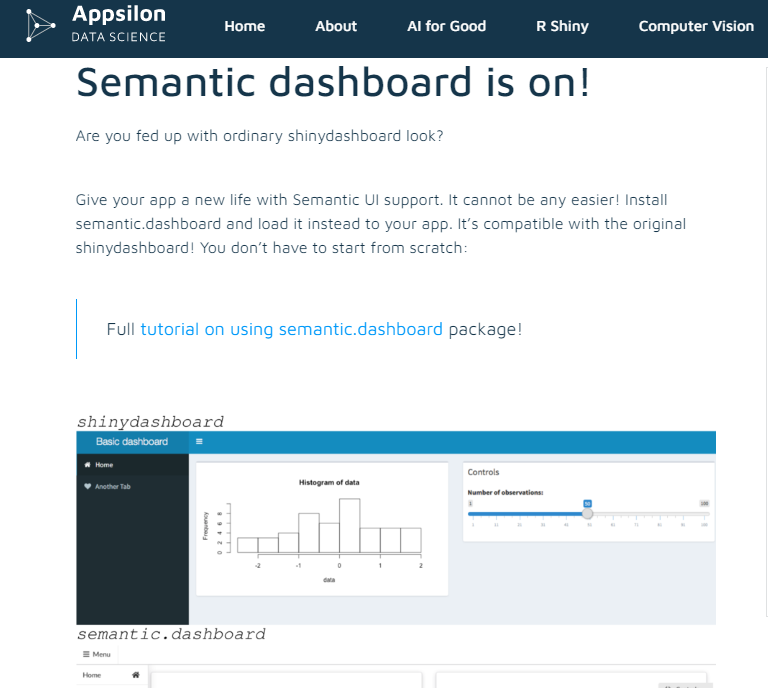
* Streamlit (<https://www.streamlit.io/>): This was an interesting option with some useful YouTube based tutorials.

*Figure: Streamlit*



* Shiny R Semantic Dashboard (<https://appsilon.com/semantic-dashboard-new-open-source-r-shiny-package/>): I had written part of CA1 for Advance Data Analytics using a Shiny R application and found these dashboard libraries during that experience. I was therefore keen to return to this toolkit for the final project.

*Figure: Shiny R Semantic Dashboard*



# Requirements: Specification and Design

## Project Mission Statement

What is the goal of this project? What business objective does it attempt to achieve?

This project intends to deliver a working system to allow a user assess a given single credit card transaction and obtain a prediction as to whether the transaction is likely to be fraudulent.

A picture containing computer

Description automatically generated

## Project Requirements

In order to achieve the objectives of the project mission statement, the following requirements must be met;

* A predictive model for Credit Card fraud detection must be built using an effective Machine Learning workflow process, which produces results that are as accurate as reasonably possible.
* A dataset is provided with sufficient volume and richness of attributes to allow for appropriate data preparation and modelling to be executed.
* All development and system execution is conducted on cloud based platforms. There is no dependency on local PC libraries or IDEs, and so on;
* All model development and deployment is conducted through the cloud based Azure Machine Learning Studio platform.
* The resultant model is accessible by a separate R Shiny application, which is also hosted online, as described in the article13 by A Kipp, 2017.
* The end user will work with the R Shiny application interface, built using RStudio Cloud, and chose a given single credit card fraud transaction to investigate. A real-time prediction of the likelihood of fraud will be provided to the use on screen.
* The R Shiny application can access the source dataset to provide data visualisations as a peripheral service to the end user.

Non-functional requirements for the system can be summarized as;

* Fully cloud based development and deployment, as mentioned above.
* The response time for real-time fraud prediction is within a 2 – 5 second timeframe.

## Project Architecture Diagram

*Figure: High Level Application Architecture Diagram*

A close up of a map

Description automatically generated

## Project Design

My system design and implementation approach follow general AGILE methodologies, as described in J. Highsmith’s 2007 book14 *Agile Project Management*. Some processes have been adapted to be practical for a project of this type.

The essence of my implementation approach is an iterative design, deploy, and assessment process.

One key tenant of the AGILE Manifesto is ‘*Working software over comprehensive documentation*’. (Beck, et al., 2001)15

Therefore, my project software is designed, coded, tested, and deployed in small discrete ‘User Stories’. (Agile Alliance, n.d.)16

I assess a ‘demonstration’, which is admittedly just to myself, at the end of one or two User Stories and then take the key learnings and observations into the next User Story.

(In the ‘real world’ I would be following a more traditional SCRUM approach, as described in the 2016 article17 by S.Sachdeva, of multiple User Stories within a pre-defined ‘SPRINT’, but I am being flexible with my interpretation of AGILE frameworks to fit with a one-person project of this type).

Section 5 of this document describes project progress to date but also concludes with a breakdown of my project plan into the constituent User Stories.

Although each User Story is refined by the preceding one it is still possible to create a general ‘roadmap’ for this project, based on the current expectations for each User Story.

For the purpose of this Interim Report I have provided the following details on the current content of the User Stories for this project.



### User Story Structure

The User Stories for this project follow a consistent format.

Each User Story described a role (or ‘actor’) and how they interface with the application within this User Story.

As each User Story iterates through the project development lifecycle the roles and actions adapt.

Each User Story has a ‘goal’. This is measured by the ‘Acceptance Criteria’ for each role, which is assessed during the demonstration of working software at the end of the completion of each User Story.

*Figure: Format of a ‘User Story’*

**Title:**

**Priority:**

**Estimate:**

**As a** *<type of user>*

**I want to** <*perform some task>*

**so that I can** <*achieve some goal>*

**Acceptance criteria**

**Given** *<some context>*

**When** <*some action is carried out>*

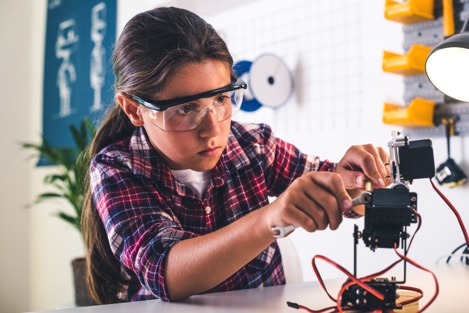
**Then** <*a set of observable outcomes should occur>*

### User Story Roles

Most User Stories in this project reflect the requirements of different roles, or system actors, and how these roles combine to deliver the end system.

The key roles within this project application are;

* **Fraud Investigator**: the end use who interacts with the Shiny R application to determine if a given credit card transaction is expected to be fraudulent.
* **Data Engineer**: the person responsible for the set-up of the Machine Learning development environment and the subsequent deployment of a Credit Card fraud model accessible by the Fraud Investigator.



* **Data Scientist**: the person who follows a Machine Learning workflow to prepare the Credit Card Fraud dataset for modelling, and who implements the most efficient modelling algorithm(s).



(Obviously in this project I shall be occupying all three roles but at different times during build, test and deployment.)

### User Story 4 – Initial Data Modelling and User Interface ‘Shell’.

User Stories 1, 2, and 3 related to the submission of the original proposal report and review with project supervisor. The key project delivery milestones are captured from User Story 4 onwards.

Acceptance Criteria become increasingly high level for the later User Stories. At this point I am building a ‘Product Backlog’ of requirements that are less and less granular in the later User Stories. This is done deliberately to allow refinements to be introduced after the assessment of the ‘demo’ as each User Story is completed/delivered.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | Access a basic Shiny R Application | See some elementary visualisations on the Credit Card Fraud dataset used for modelling.  See a basic screen that shows a credit card transaction invoking a Fraud prediction algorithm and producing an output. |
| 1.2 | Data Engineer | Set up an Azure Machine Learning Studio Environment. | Provide a Machine Learning development environment for the Data Scientist. |
| 1.3 | Data Engineer | Deploy a basic REST API endpoint for a Fraud prediction model. | Provide access from an external program to the Credit Card Fraud predictive model. |
| 1.4 | Data Scientist | Build a basic Fraud Prediction model with 2.5K rows of Credit Card fraud data. | Quickly validate Azure Machine Learning Studio as means to generate a model for use in the Shiny R application prototype. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | An RStudio environment | Investigator opens Shiny R App | Basic dataset information is available.  A hard-coded example of the output result of the API calling the Fraud model is displayed. |
| 1.2 | Access to Azure Account / Workspace | Data Scientist creates a new Machine Learning Studio Azure Workspace | All the required data storage, ML Designer, and REST Endpoint tools are available. |
| 1.4 | Access to Azure Machine Learning Studio. | Data Scientist logs into Azure Machine Learning Studio. | Data Scientist can navigate interface and generate initial basic predictive Fraud model. |
| 1.3 | A basic Fraud predictive model has been developed by the Data Scientist. | The model is deployed as a Web Service in Azure. | The API to the Fraud model can be accessed externally from another system (Shiny R App). |

### User Story 5 – Shiny App Prototype

Although User Story 4 provides for a basic Shiny R App, there is relatively little interactive functionality for the user, apart from basic shiny Desktop navigation features.

User Story 5 enriches the Shiny R App user interface.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | Select a given ‘new’ Credit Card transaction and assess likelihood of Fraud. | Choose and individual Credit Card transaction from a pre-loaded list and involve the predictive fraud model. |
| 1.2 | Data Engineer | Refine Shiny R app to routines to parse attributes of a user selected CC transaction. | Allow for interactive calls on API for predictive Fraud model. |
| 1.3 | Data Scientist | Refine Shiny R app to routines to display more sophisticated view of data used for modelling. | Give the Investigator (and Scientist) a better view of the source CC transactions used for modelling. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | A pre-loaded lists of CC transactions are provided for the Investigator. | Investigator : Selects a given CC transaction | API calls returns result of predictive Fraud model : is transaction likely to be fraudulent? |
| 1.2 | Interactive selection of a CC transaction. | API call is made in code. | Parameters in API call are dynamically updated with user CC trxn selection. Involving API returns an on screen ‘score’ to show likelihood of fraud. |
| 1.3 | Access to Azure hosted dataset used for modelling. | R Shiny App launches. | Data Scientist provides routines in R to display meaningful graphical representations of CC dataset. |
| 1.4 | R Shiny App loading dataset for visualisation and invoking Fraud API. | The R Shiny App reads data from Azure datastore and REST Endpoints | Performance is acceptable (2 – 5 seconds) with no on-screen errors. |

### User Story 6 – Hosted Prototype

User Story 5 improved the richness of the application UI but it still runs as an application through RStudio Cloud. Although it is not necessarily running on a ‘local’ system, the next User Story moves the application onto a true hosted platform where it can be run independently.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | Log onto a hosted location for the R Shiny App | Run the application without the need to execute the app within RStudio Cloud |
| 1.2 | Data Engineer | Deploy the application onto a ShinyIO platform | Verify connectivity to the Azure hosted data and REST Endpoint for the Fraud model from outside RStudio Cloud. |
| 1.3 | Data Scientist | Compare output of data modelling between RStudio Cloud and ShinyIO deployment | Confirm that the predictive Fraud model results are consistent, regardless of platform. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | A ShinyIO web location provided by the Data Engineer. | Investigator : Selects the new web location for the application. | The UI experience is identical to the RStudio Cloud based Shiny R App. |
| 1.2 | Interactive selection of a CC transaction. | API call is made in code. | Results are consistent for any given CC trxn when RStudio Cloud and ShinyIO apps are compared. |
| 1.3 | ShinyIO : R Shiny App loading dataset for visualisation and invoking Fraud API. | The R Shiny App reads data from Azure datastore and REST Endpoints | Performance is acceptable (2 – 5 seconds) with no on-screen errors. |

### User Story 7 – Feature Enhancements 1

Following AGILE Manifesto principles, the User Stories so far have focused on a constant iterative delivery of working software, primarily through the R Shiny Application interface.

User Story 7 returns to the core Machine Learning objectives of the project and the need to refine and improve the predictive Fraud model, which will be used in the final version of the project application.

The predictive model used to date in the project lifecycle was created quickly for the purposes of end-to-end testing and early deployment validations. However, it is now necessary to apply more robust and thorough Machine Learning techniques to the final model that I will deploy into Production.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | Select CC transactions from external files, not just preloaded data. | Predict Fraud against a more interactive range of file based CC transactions. |
| 1.2 | Data Scientist | Perform data manipulation routines on a larger CC dataset as part of re-modelling iterations. | Build more efficient and accurate predictive Fraud models, using techniques such as feature selection etc. |
| 1.3 | Data Scientist | Perform analysis of the CC dataset with multiple algorithms. | Determine the best model, with supporting documentation, for productive modelling on this CC dataset. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | An updated ShinyIO web UI provided by the Data Engineer. | Investigator : Selects an external file of CC transactions through the UI. | The Shiny R App can read the transactions and allow for a selective fraud analysis on any given entry. |
| 1.2 | The available Machine Learning Studio Web Designer Data Tools. | A larger CC dataset is loaded. | The dataset output has been manipulated to:  - focus on the most critical features  - handle any non-numeric values  - standardise any numeric data  - any other relevant data transformation, all of which are documented |
| 1.3 | The available Machine Learning Studio Web Designer Modelling Tools. | Data Scientist: iterates through suitable algorithms | A more accurate model is generated backed up by documented comparisons with earlier algorithms. |

### User Story 8 – Feature Enhancements 2

User Story 7 refines the quality of the CC Fraud predictive model. User Story 8 redeploys this model through the Shiny R Application and adds improvements to the UI.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | See enhanced data visualisations on the source CC dataset (used by the modelling process). | Obtain a more sophisticated view of the key attributes of the CC Fraud trxn data. |
| 1.2 | Fraud Investigator | See an onscreen display of the key data elements of a selected CC transaction before it is submitted to the Fraud model for assessment. | Have a better view of the key data elements being passed to the fraud predictive model. |
| 1.3 | Data Engineer | Re-deploy the revised REST Endpoint API for the revised predictive Fraud model build by the Data Scientist. | Provide the application with access to the improved fraud predictive model. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | An updated ShinyIO web UI provided by the Data Engineer. | Investigator: Selects initial Shiny R application dashboard tab that contains data visualisations. | The Shiny R App displays an improved set of visualisations, based on recommendations from Data Scientist. |
| 1.2 | An updated ShinyIO web UI provided by the Data Engineer. | Investigator: Selects an external file of CC transactions through the UI. | The Shiny R App displays a description of the key attributes in the CC transaction (based on directions from the Data Scientist). |
| 1.3 | New model provided by the Data Scientist. | Engineer: deploys the new API routines in the R Shiny Application. | The R Shiny Application continues to return predictive outputs for the Fraud model (when compared to test results in the Azure Machine Learning Studio). |

### User Story 9 – Final Project Refinements

This User Story has the least granular set of requirements and is intended as a placeholder for refinements that arise out of the ‘demonstration’ of working software after the delivery of User Stories 7 and 8.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User Role / Objective** | | |
| **USER STORY ID** | **As a** *<type of user>* | **I want to** *<perform some task>* | **so that I can** *<achieve some goal>* |
| 1.1 | Fraud Investigator | See revisions to the Shiny R Application UI based on feedback from US 7 and US 8. | Declare acceptance of the project as fit for ‘Production’, which is the final presentation of the project in DBS. |
| 1.2 | Data Scientist | Implement revisions to the CC predictive fraud model based on feedback from US 7 and US 8. | Declare acceptance of the predictive Fraud model as fit for ‘Production’, including all supporting documentation. |
| 1.3 | Data Engineer | Implement revisions to the fraud model deployment based on feedback from US 7 and US 8. | Declare acceptance of the deployment of the Fraud model as fit for ‘Production’, including all supporting documentation. |
|  | **Acceptance Criteria** | | |
|  | *Given <some context>* | **When** *<some action is carried out>* | **Then** *<a set of observable outcomes should occur>* |
| 1.1 | An updated ShinyIO web UI provided by the Data Engineer/Data Scientist. | Investigator: Launches the Shiny R application. | The Shiny R App runs and meets all the expected requirements for a final demonstration. |

# Project Testing and Evaluation

## Evaluation of User Stories

The User Stories in Section 3.3 describe the functional and non-functional requirements of the application.

At the end of each User Story, I will have developed a further iteration of working software for the application.

The ‘Acceptance Criteria’ provided within each User Story provide a series of checkpoints against which I assess the success of the User Story in meeting system requirements.

If requirements are not met, then the following User Story is adjusted to incorporate any changes that must be carried forward into the project development lifecycle.

A picture containing device

Description automatically generated

## Project Testing

Is there a formal QA test cycle within this project? Yes, but it is not a ‘Waterfall’ activity at the end of a monolithic phase of development.

Incorporated into each User Story is the assumption that work included design, development, deployment, **and** testing.

What is the criteria, or guidance, for testing? Each User Story has an inherent ‘**Definition of Done**’, as described in an 2017 article18 by D.Huether, that must be met before the User Story is considered fit for purpose and the software can fairly be described as ‘working’.

The Definition of Done (DoD), which I apply to each User Story is;

* Each User Story must generate a new iteration of working software, which is a functional enhancement on the previous User Story.
* All Acceptance Criteria in the User Story are met in full.
* No critical defects remain at the end of the User Story activity.
* Minor or cosmetic defects can be carried over to the next User Story if appropriate but must be addressed in that User Story.
* Screen refreshes when data is loading, or the Fraud API is being invoked, should require no more than 2 – 5 seconds. Usability design should try and minimise data loads when the user first enters a Shiny dashboard tab.
* A documented one-page test report is produced at the end of the User Story. These test reports will be included in the Final Report for the project.

A User Story must pass the DoD. If it does not, then the next User Story cannot start. Work on the User Story is extended until the DoD are met.

**A close up of a device

Description automatically generated**

# Demonstration of Progress

## Credit Card Fraud Application: Prototype Location

The prototype for this project application is currently hosted on *shinyapps.io* and the UI can be accessed through this URL;

<https://ciaran-finnegan.shinyapps.io/DBS_CCFraudRShinyApp_10524150/>

## Credit Card Fraud Application: User Guide (Prototype)

The prototype is intended to demonstrate progress to date and provide tangible evidence of the end goals of this project.

*Figure: Project Prototype*

A screenshot of a cell phone

Description automatically generated

A User Guide, in Microsoft PowerPoint format, is embedded with this report, and has also been submitted separately.



This provisional guide provides a walkthrough of the key functional aspects of the prototype project artefact.

## Project Plan 2020: Current Status – 14th August 2020

**(Produced using the Team Gantt online portal)**

A screenshot of text

Description automatically generated

# Project Re-Design Considerations

## Initial Proposal

The User Stories described in Section 3.3 of this Interim Report are an extrapolation of the system described in the initial project proposal, as submitted on Friday June 26th 2020.

In terms of high-level functionality, I have no plans to radically deviate from these proposals at this point in the project lifecycle.

## Possible Future Design/Deployment Considerations

Certain implementation options were left open for consideration in terms of the build of this project. All such options would be dependent on remaining timeframe and complexity.

My objective is to deliver the project as described in Section 3 of this report but I have some flexibility to adapt the project in the following areas;

* The Azure Machine Learning ***Classic*** Studio is the development environment of choice for this project. I have experimented with the more up to date Azure Machine Learning workspace with Pipelines, based on a 2019 tutorial19 from Tim Verner, but this is notably more complex to work within and a more expensive platform on which to deploy. If I encounter functional limitations in the Azure Classic workspace then I may need to consider switching development environments.
* Data manipulation and modelling will be implemented through the Azure Machine Learning ***Classic Designer***. This does allow for embedded Python and/or R routines but is essentially a graphical ‘no-code’ tool, not unlike RapidMiner or other such products. If this proves too limiting, or I have time to experiment further, I may deploy the revised predictive Fraud model through a Python *Jupyter Notebook*. I have run a small number of trial Python Notebook experiments and, as above, the issue is mainly with complexity and cost to deploy a REST Endpoint.
* The UI for my project is being built as a Shiny R Application using the open source Semantic Dashboard libraries. This is unlikely to change. However, should I encounter issues with a full hosting arrangements I may consider switching to a Python based web application written using the *Streamlit* libraries (Deliwala, 2019)20. This Streamlit option would avoid the complexities of considering *Django* and *Flask* but would only be considered if I encounter serious issues with the deployment of my Shiny R Application for the system UI.

# Appendices

## Azure Generated Code Segments

The Azure Machine Learning Studio auto-generates codes segments in C#, Python 3.6, and R to access both Azure hosted datastores and invoke APIs to Azure hosted Rest Endpoints for deployed models.

Below are examples of code snippets which have been incorporated into my project.

This code segments reads a subset of the Credit Card dataset and uses the output to generate data visualisations in the R Shiny App.

*Figure: R code snippet to read Azure hosted datastore*

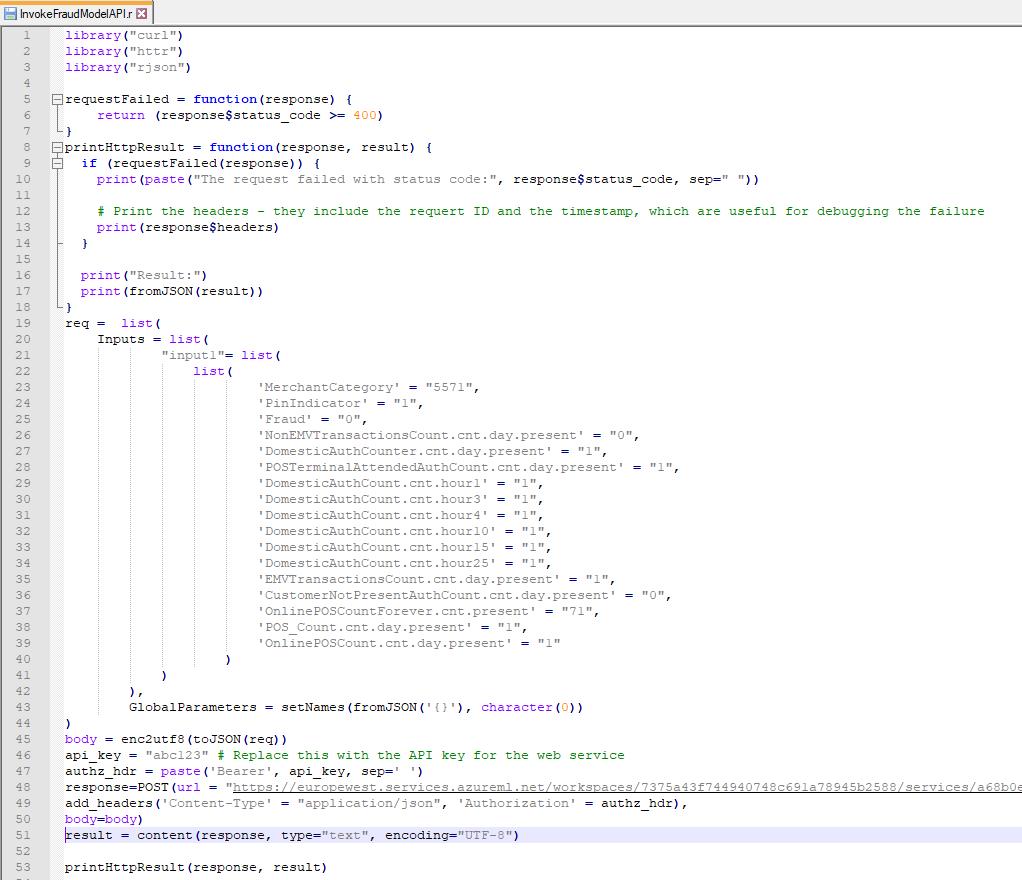
A screenshot of a social media post

Description automatically generated

This code segment invokes the API to pass attributes to the Fraud detection model, hosted in Azure, and returns a prediction score.

(Line 48 has been truncated slightly).

*Figure: R code snippet to read Azure hosted REST Endpoint for Fraud Model*



## Shiny R Application Code Files

*Figure: RStudio Cloud Environment*

A screenshot of a social media post

Description automatically generated

Attached is a zip file that contains the current bespoke R code that I have written for my project. (This is obviously a work in progress and includes commented out lines of code and some redundant files).

The code is modularised and broken into sub folders based on function.



The final report for the project will contain a comprehensive view of the application code structure.

## Azure Machine Learning Classic Studio Experiments

The final versions of the model used to provide the predicted fraud score for ‘new’ transactions will be built during the User Stories described in Sections 3.3.6 and 3.3.8 of this document.

In order to quickly build an end-to-end Fraud detection prototype the initial modelling exercise has been relatively straightforward, and the experiment is represented in the diagram below.

*Figure: Initial Logistic Model for Fraud Prediction*

A screenshot of a cell phone

Description automatically generated

*Figure: WebService Generated in Azure ML Studio Classic (for deployment)*

A screenshot of a social media post

Description automatically generated

The Feature Selection module was a manual selection of explicit columns based on external advice from a company colleague who has prior industry experience in the area of payment fraud.

User Story 7 will involve an exploration of Machine Learning techniques to perform a more formal, and automated, feature selection process.

## Credit Card Fraud Datasets

The dataset I am using is a large volume of data and too unwieldy to include in this document.

A significant representative volume is stored in Azure at this stage in the project but requires an Azure account to access.

For illustrative purposes, I have attached a much-reduced version of the dataset with the full set of columns but only 200 rows.



The prototype application loads 5 records that have already been manipulated through a manual feature selection process. That dataset is stored on Azure but is very small and is included here in this report also.



# References / Bibliography - Interim Report

1Microsoft Tutorial, 2020. *What Is ML Studio (Classic) - Azure*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/machine-learning/studio/what-is-ml-studio> [Accessed 19 June 2020].

2Krzemiński, D., 2018. *Create Outstanding R Shiny Dashboards With The Semantic.Dashboard Package - Appsilon Data Science | End­ To­ End Data Science Solutions*. [online] Appsilon Data Science | End­ to­ End Data Science Solutions. Available at: <https://appsilon.com/create-outstanding-dashboards-with-the-new-semantic-dashboard-package/> [Accessed 28 June 2020].

3Parsons, T., 2019. *When To Use Waterfall Vs. Agile | Macadamian*. [online] Macadamian. Available at: <https://www.macadamian.com/learn/when-to-use-waterfall-vs-agile/> [Accessed 10 July 2020].

4Stanton, K., 2019. *Card Fraud In Europe - It's Still Increasing..*. [online] Finextra Research. Available at: <https://www.finextra.com/blogposting/16824/card-fraud-in-europe---its-still-increasing> [Accessed 4 August 2020].

5Cherowbrier, J., 2019. *Forecast Of Card Payments Per Day 2006, 2016 And 2026 Kingdom | Statista*. [online] Statista. Available at: <https://www.statista.com/statistics/719708/card-payments-per-day-forecast-united-kingdom/> [Accessed 7 August 2020].

6ICA, n.d. *What Is Customer Due Diligence (CDD)? | ICA*. [online] Int-comp.org. Available at: <https://www.int-comp.org/careers/your-career-in-aml/what-is-customer-due-diligence-cdd/> [Accessed 7 August 2020].

7Machine Learning Group - ULB, 2018. *Credit Card Fraud Detection*. [online] Kaggle.com. Available at: <https://www.kaggle.com/mlg-ulb/creditcardfraud> [Accessed 13 August 2020].

8Preda, G., 2018. *Credit Card Fraud Detection Predictive Models*. [online] Kaggle.com. Available at: <https://www.kaggle.com/gpreda/credit-card-fraud-detection-predictive-models#Data-exploration> [Accessed 13 June 2020].

9Charminda D, 2020. *Credit Card Fraud Detection: A Detailed Study*. [online] Kaggle.com. Available at: <https://www.kaggle.com/chandrimad31/credit-card-fraud-detection-a-detailed-study> [Accessed 13 July 2020].

10Escapa, C., 2018. *AWS Analytics Services Explained: From Data Lakes To Machine Learning | Amazon Web Services*. [online] Amazon Web Services. Available at: <https://aws.amazon.com/blogs/apn/aws-analytics-services-explained-from-data-lakes-to-machine-learning/> [Accessed 7 August 2020].

11Sharma, A., 2020. *How To Use Google Colab For Deep Learning And Machine Learning*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/#1> [Accessed 7 August 2020].

12Microsoft Tutorial, 2020. *ML Studio (Classic): Quickstart: Create A Data Science Experiment - Azure*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/machine-learning/studio/create-experiment> [Accessed 7 August 2020].

13Kipp, A., 2017. *Shiny - Shinyapps.Io - Getting Started*. [online] Shiny.rstudio.com. Available at: <https://shiny.rstudio.com/articles/shinyapps.html> [Accessed 7 July 2020].

14Highsmith, J., 2007. *Agile Project Management*. 23rd ed. Boston, Mass.;Munich[u.a.]: Addison-Wesley.

15Beck, et al., 2001. *Manifesto For Agile Software Development*. [online] Agilemanifesto.org. Available at: <https://agilemanifesto.org/> [Accessed 7 August 2018].

16Agile Alliance, n.d. *What Are User Stories?*. [online] Agile Alliance. Available at: <https://www.agilealliance.org/glossary/user-stories> [Accessed 7 March 2018].

17Sachdeva, S., 2016. Scrum Methodology. *International Journal Of Engineering And Computer Science*,.

18Huether, D., 2017. *The Definition Of Done*. [online] LeadingAgile. Available at: <https://www.leadingagile.com/2017/02/definition-of-done/#:~:text=The%20definition%20of%20done%20(DoD,%2C%20team%2C%20or%20consuming%20system.> [Accessed 7 March 2018].

19Verner, T., 2019. *Microsoft Azure AI Engineer: Developing ML Pipelines In Microsoft Azure*. [online] Pluralsight.com. Available at: <https://www.pluralsight.com/courses/microsoft-azure-developing-ml-pipelines> [Accessed 7 June 2020].

20Deliwala, S., 2019. *Streamlit 101: An In-Depth Introduction*. [online] Medium. Available at: <https://towardsdatascience.com/streamlit-101-an-in-depth-introduction-fc8aad9492f2> [Accessed 7 August 2020].