

Predictive Analysis and deployment for CREDIT CARD FRAUD Detection using Machine Learning techniques

**Final 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: 25th September 2020

**Abstract**

In 2016, the total value of fraud for cards issues within the SEPA region was €1.8 billion (Whatman, 2019). The company with which I am employed works in the domain of Financial Crime software, so I choose to focus this project on one of the key challenges in this area. My project delivers an end-to-end solution that uses an Azure hosted predictive model, accessed via a separate Shiny R dashboard, to assess individual credit card transactions in real time for the likelihood of fraud.

A dataset of 25K+ historical US credit card transactions (2104), each one labelled as ‘Fraud’ or ‘Not Fraud’, is engineered to train and deploy a model to predict if ‘future’ transactions appear to be fraudulent.

The Machine Learning modelling process is managed through the online Microsoft Azure Machine Learning Studio (classic) platform. This includes the hosting of a REST Endpoint for the model, to be accessed as a Web Service by an external application for fraud prediction.

A separate Shiny R dashboard is included in this project to access the predictive fraud model through an API call, passing the details of ‘new’ card transactions as parameters one-by-one in real time to the Azure Web Service.

This Shiny dashboard is hosted in in the ShinyIO platform and also provides a secondary interface to provide key data visualisation graphs on the original dataset

**Acknowledgments**

I want to acknowledge the advice and support provided by two fellow work colleagues from BAE Systems Applied Intelligence, 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 Final Report and the with the wider project in general.

# Contents

[Contents 2](#_Toc51436283)

[1. Introduction 4](#_Toc51436284)

[1.1. What the Project Aimed to Deliver 4](#_Toc51436285)

[1.2. How the Project Delivery was Implemented 4](#_Toc51436286)

[2. Background / Literature Review 5](#_Toc51436287)

[2.1. Credit Card Fraud Detection: Further Research on Predictive Models 5](#_Toc51436288)

[2.2. Credit Card Fraud Detection: In Context – My Dataset 7](#_Toc51436289)

[3. Requirements: Specification and Design 8](#_Toc51436290)

[3.1. High Level Project Requirements 8](#_Toc51436291)

[3.2. Project Architecture Diagram 9](#_Toc51436292)

[3.3. High Level Project Design 10](#_Toc51436293)

[3.3.1. Prototype Development – Initial User Stories 10](#_Toc51436298)

[3.3.2. Final Project Deliverable – Further User Stories 10](#_Toc51436299)

[4. Project Implementation (1) – Azure Modelling 11](#_Toc51436304)

[4.1. The Machine Learning Workflow 11](#_Toc51436305)

[4.2. Credit Card Fraud – The Azure Workspace/Machine Learning Studio 13](#_Toc51436306)

[4.3. Credit Card Fraud Dataset – Analysis and Preparation 16](#_Toc51436307)

[4.3.1. Experiment 1: Data Cleansing 17](#_Toc51436308)

[4.3.2. Experiment 2: Feature Engineering 18](#_Toc51436309)

[4.3.3. Experiment 3: Feature Selection 19](#_Toc51436310)

[4.4. Credit Card Fraud – Building the Azure Model 20](#_Toc51436311)

[4.4.1. Experiment 4: Basic Model Evaluation with Feature Engineering 21](#_Toc51436313)

[4.4.2. Experiment 5: Model Evaluation with Cross Validation/Hyperparameter Tuning 23](#_Toc51436314)

[4.4.3. Experiment 6: Comparison of Multiple Classification Algorithms (1) 25](#_Toc51436315)

[4.4.4. Experiment 7: Comparison of Multiple Classification Algorithms (2) 27](#_Toc51436316)

[4.5. Credit Card Fraud – Deploying the Azure Model 28](#_Toc51436317)

[4.5.1. Experiment 8: Feature Engineering on Larger Dataset 29](#_Toc51436319)

[4.5.2. Experiment 9: Creation of Predictive Fraud Model for Deployment 30](#_Toc51436320)

[4.5.3. Deployment and Validation of Web Service for Predictive Fraud Model 33](#_Toc51436321)

[5. Project Implementation (2) – Shiny R Dashboard UI 35](#_Toc51436322)

[5.1. Data Visualisations in a Shiny Dashboard 35](#_Toc51436323)

[5.2. Credit Card Fraud – UI to Check Fraud Predictions 35](#_Toc51436324)

[5.3. Shiny UI – Hosted Application 35](#_Toc51436325)

[6. Testing and Results 36](#_Toc51436326)

[6.1. User Story ‘Demos’ – Test Results and ‘Feedback’ 36](#_Toc51436327)

[6.1.1. User Story 4: Initial Data Modelling – Review and Evaluation 36](#_Toc51436331)

[6.1.2. User Story 5: Basic Shiny App – Review and Evaluation 37](#_Toc51436332)

[6.1.3. User Story 6: Integrated Prototype – Review and Evaluation 38](#_Toc51436333)

[6.1.4. User Story 7: Enhanced Modelling – Review and Evaluation 39](#_Toc51436334)

[6.1.5. User Story 8: Enhanced UI – Review and Evaluation 40](#_Toc51436335)

[6.1.6. User Story 9: Presentation Preparation – Review and Evaluation 41](#_Toc51436336)

[6.2. Final Project Assessment: A Critical Evaluation 42](#_Toc51436337)

[6.3. Project Plan 2020: Final Status – 25th September 2020 43](#_Toc51436338)

[7. Project Location and User Guide 44](#_Toc51436339)

[7.1. Credit Card Fraud Application: Prototype Location 44](#_Toc51436340)

[7.2. Credit Card Fraud Application: User Guide (Final Project) 44](#_Toc51436341)

[8. Project Conclusions 45](#_Toc51436342)

[8.1. Where Project Goals Achieved? 45](#_Toc51436343)

[8.2. Future Design/Deployment Considerations 46](#_Toc51436344)

[9. Appendices 48](#_Toc51436345)

[9.1. Azure Generated Code Segments 48](#_Toc51436346)

[9.2. Shiny R Application Code Files 50](#_Toc51436347)

[9.2.1. Diagram: The RStudio Cloud Environment 50](#_Toc51436353)

[9.2.2. The Shiny UI Code – Data Visualisations 51](#_Toc51436354)

[9.2.3. The Shiny UI Code – Transaction Fraud Detection 51](#_Toc51436355)

[9.2.4. The R Code Parsing Data and Invoking UI 51](#_Toc51436356)

[9.3. Azure Machine Learning Classic Studio Experiments 52](#_Toc51436357)

[9.3.1. Experiment 1: Breakdown 52](#_Toc51436359)

[9.3.2. Experiment 2: Breakdown 57](#_Toc51436360)

[9.3.3. Experiment 3: Breakdown 61](#_Toc51436361)

[9.3.4. Experiment 4: Breakdown 64](#_Toc51436362)

[9.3.5. Experiment 5: Breakdown 66](#_Toc51436363)

[9.3.6. Experiment 6: Breakdown 68](#_Toc51436364)

[9.3.7. Experiment 7: Breakdown 69](#_Toc51436365)

[9.3.8. Experiment 8: Breakdown 70](#_Toc51436366)

[9.3.9. Experiment 9: Breakdown 71](#_Toc51436367)

[9.4. Credit Card Fraud Datasets 74](#_Toc51436368)

[10. References / Bibliography – Final Report 75](#_Toc51436369)

# Introduction

## What the Project Aimed to Deliver

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.

The project user interface is a Shiny R Dashboard hosted on the *Shinyapps.io* platform. See Section 7 of this document for the URL and User Guide.

The predictive model itself has been built in an *Azure Machine Learning Classic Studio Workspace*. See Section 9.3 of this document for a detailed description of the Machine Learning workflow process employed to engineer the dataset, train and evaluate the model, and then deploy the Web Services to allow access to the predictive model.

## How the Project Delivery was Implemented

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 predictive fraud model was built and deployed using the following steps:

* My credit card fraud dataset contains 25K rows, each containing a label for ‘fraud’.
* A dedicated project has been used for all the project artifacts; datasets, experiments, models, Web Services etc.
* A sequence of experiments in Azure ML Studio (classic) is used on Feature Engineering of the dataset prior to modelling.
* Various classification algorithms are evaluated, leading to further iterations of the Feature Engineering experiments.
* A Model is trained based with the most optimal model, using the final Feature selection decisions. This model is deployed as an Azure hosted Web Services.

# Background / Literature Review

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

Section 2.2 of the Interim Report on this project elaborated on two Kaggle submissions made in related to the credit card fraud dataset generated by the work of the Machine Learning Group (<http://mlg.ulb.ac.be>) of ULB (Université Libre de Bruxelles).

References in both those submissions referred to ongoing studies in the area of credit card fraud detection that are being collected by the ResearchGate network for scientists and researchers.

*Figure: www.researchgate.net/project/Fraud-detection-with-machine-learning*

A screenshot of a cell phone

Description automatically generated

The latest submission, as of September 2020, contains an interesting paper of credit card fraud detection with a focus on transaction sequence. However, the initial sections of this submission (Lucas et al., 2019)**n** provide an excellent overview of the challenges facing credit card detection in the real world and solutions that have emerged over the last 10+ years.

Reading through this material I have drawn on key observations to direct my work on this project.

*Algorithm Selection*

A paper from Bhattacharyya, Jha, Tharakunnel and Westland, 2011**n** described research on a real-world US credit card dataset. It involved a comparison of Support Vector Machine, Random Forest, and Logistic Regression, which – as expected - are all algorithm options I have access to in Azure ML Studio (classic).

There are many observations in the paper but some key points I took note of (and are repeated in other articles) were:

* Credit card data is often very imbalanced. Fraud can be disastrous when it happens but it a tiny proportion of overall records. A defined sampling approach is a definite requirement.
* The Fraud/non-Fraud imbalance can make the use of ‘Accuracy’ in a Confusion Matrix as somewhat ineffective.
* Accurate identification of fraud is often a primary requirement so there is a need to look at the trade-offs in improving Recall and Precision.
* Logistic Regression can perform consistently well but is dependent on the approach to Feature Engineering.

One interesting note came from other research is that the imbalanced nature of credit card data makes fraud a candidate for anomaly detection routines (Ceronmani Sharmila et al., 2019)**n**. Algorithms such as ‘Isolated Forest’ or ‘Local Outlier Factors (LOF)’ are frequently recommended.

As I explain in Section 2.2 of this document, I choose not to adopt these unsupervised approaches and focused much of my time in the Machine Learning workflow for this project on Feature Engineering leading into supervised learning techniques.

*Feature Engineering*

Although I was not able to read all the details in the paper by Mahmoudi and Duman, 2015**n** on fraud detection analysis, several commentators on this study referred to the benefit of being able to work with the ‘raw’ features of a credit card dataset.

This is an advantage I have with my dataset, as opposed to the previously mentioned ULB dataset, which is heavily anonymised through PCA.

However, the opening lines of a paper from Lima and Pereira, 2017 on ‘*Feature Selection Approaches to Fraud Detection in e-Payment Systems*’ states that “..*Due to the large amount of data generated in electronic transactions, to find the best set of features is an essential task to identify frauds*.“

Given that my starting dataset has 380 columns, this was a guiding principle for me.

I was also going to have to code in R to invoke an API to call my predictive fraud model with all the ‘important’ features on ‘new’ credit card transactions passed as parameters. Therefore, reducing the complexity of setting up this parameter list for the API code would help improve robustness of the UI code.

*Transaction Sequences*

The ResearchNet articles provided references to additional papers on how to improve Feature Engineering for fraud analysis by creating aggregates and time series analyses of the transactions. I choose not to explore this avenue because of the potential complexity.

There are many columns in my dataset that look at time since transaction but my primary actions with this data was just to remove any highly correlated features.

## Credit Card Fraud Detection: In Context – My Dataset

To give an overview of my credit card dataset:

1. It contains 25,128 rows and 380 columns.
2. This is a live dataset of North American credit card transactions from 2013. Only names and initial address lines have been anonymised. Apart from data cleansing, no other alterations to the ‘raw’ transactions have taken place.
3. Previously, the data was used for a, now discontinued, credit card fraud product that relied on a ‘Rule Engine’ to generate alerts for potential fraud.
4. The data is expected to be free of corrupt data elements, and largely free of missing data.
5. Many columns still present in the dataset were created as the result of ETL processes from other peripheral systems and are redundant. No domain knowledge in this area has been documented.
6. Approximately 16.2% of the dataset records are known fraud cases. The data has been balanced over a period of time based on the 2013 dataset. This is a significant advantage/difference from other comparable research datasets in the public domains.
7. The original project proposal, as described in the Interim Report, was to use a dataset of 280K records but that became infeasible due to reasons elaborated on in Section 8 of this document (Conclusions).
8. 10% of the dataset was used for initial feature engineering in Azure ML Studio (classic), but the full dataset was used to train the production model.

Based on the research described in the previous section (2.1), my approach to building this predictive credit card fraud model will focus on:

* Data sampling and balancing and can be relatively straightforward for my project. Other industry papers have devoted significant amounts of time to addressing the challenge of balancing a very small sub-set of actual fraud data.
* Feature Engineering will be very important. This is true of most Machine Learning problems, but I need to reduce the number of features from a starting number of 380.
* Algorithm selection, to build my predictive fraud model, can focus on binary Classification options for supervised learning.

# Requirements: Specification and Design

## High Level Project Requirements

In order to achieve the objectives of the project mission statement, as described in the Interim Report document, the following requirements needed to be met;

* A dataset is provided with sufficient volume and richness of attributes to allow for appropriate data preparation and modelling to be executed.
* A predictive model for Credit Card fraud detection is built using an effective Machine Learning workflow process, which produces results that are as accurate as reasonably possible.
* All development and system execution takes place on cloud-based platforms. There is no dependency on local PC libraries or IDEs, etc.
* 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.

## Project Architecture Diagram

*Figure: High Level Application Architecture Diagram*

A close up of a map

Description automatically generated

## High Level Project Design

The Interim report provided a detailed overview of the User Stories used to map out the design and implementation of this project.



### Prototype Development – Initial User Stories

*Figure: Initial Basic Modelling in Azure ML Studio (Classic) and Basic UI Deployment*

A screenshot of a cell phone

Description automatically generated

For the Interim Report a basic model, with limited Feature Engineering and no tuning was trained and deployed as a Web Services. The Shiny R dashboard was a hosted application but passed in pre-loaded transactions for fraud assessment.

### Final Project Deliverable – Further User Stories

*Figure: Enhanced Modelling in Azure ML Studio (Classic) and Enhanced UI Deployment*

A screenshot of a computer

Description automatically generated

The final version of the project involved a full Machine Learning workflow process to train and deploy a reliable predictive model. The user can select ‘new’ credit card transactions from multiple files and assess any given one for fraud in real time.



# Project Implementation (1) – Azure Modelling

## The Machine Learning Workflow

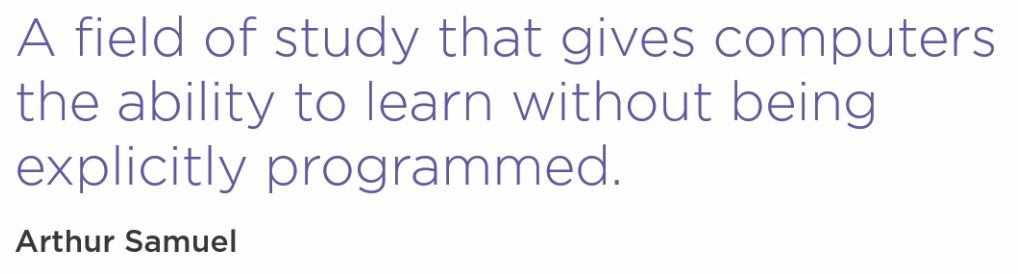
A significant amount of training and reference material I had access for guidance on using the Azure Machine Learning Studio came from Pluralsight courses.

I had reproduced a number of illustrations from those courses (and cited the sources) to;

* Explain my general approach to using Machine Learning processes to build my credit card Fraud predictive model.
* Describe how Azure Machine Learning Studio was used to implement the key steps in the Machine Learning process for this project.

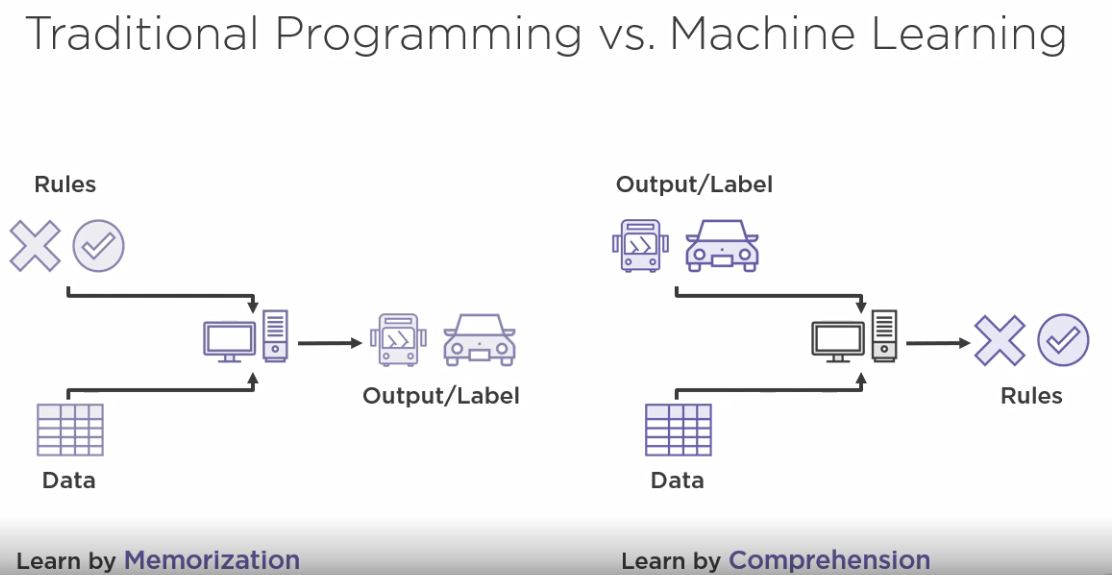
To start with a quote…’what is Machine Learning?’

*Figure: Reproduced Quote Image from Pluralsight (Kurata, 2016)*



The difference between Machine Learning and ‘traditional programming’ can be illustrated briefly as follows.

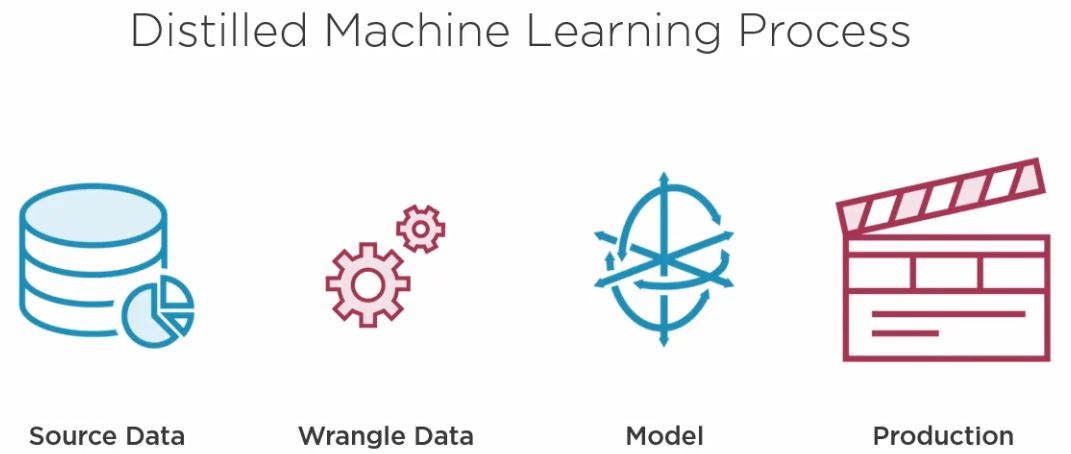
Figure: Traditional Programming v Machine Learning - *Reproduced Image from Pluralsight* (Rhodes, 2020)



This project aims to create a model that can take unseen data and determine a prediction as to whether the transaction is fraudulent, as opposed to a approach such as writing code ti implement a sequence of pre-defined rules.

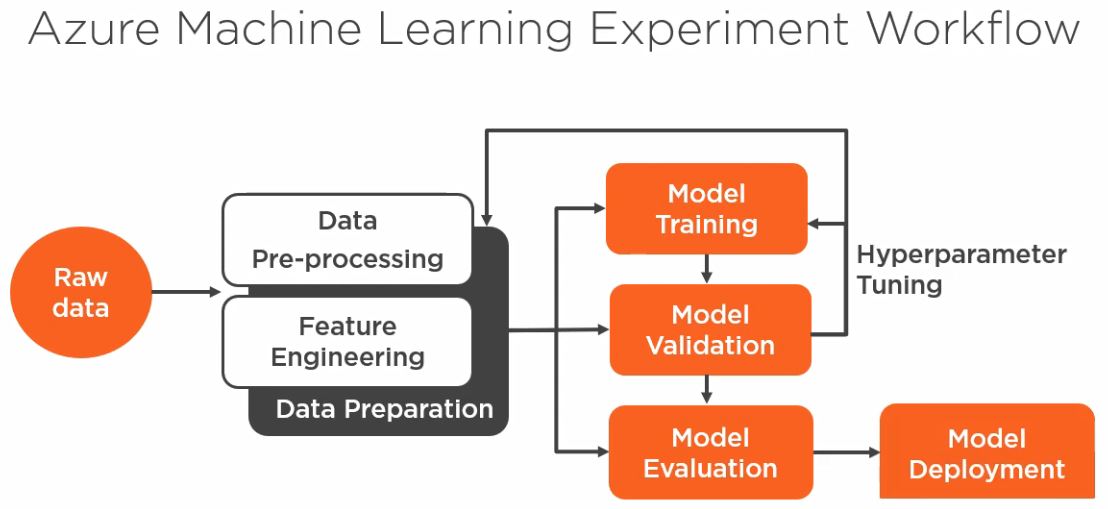
This is a simplified diagram of how the Machine Learning process is applied.

*Figure: Reproduced Image from Pluralsight (Kurata, 2016)*

**

The following figure shows the steps in Azure Machine Learning Studio about which I will provide further implementation details in Section 4.2 through to Section 4.5.

Figure: *Reproduced Image from Pluralsight* (Rhodes, 2020)



## Credit Card Fraud – The Azure Workspace/Machine Learning Studio

The steps to create an Azure account and Workspace are well documented by Microsoft, and I have not sought to reproduce them in detail in this document.

Similarly, the set up required for Azure Machine Learning Studio is equally well documented and accessible from within the Azure portal.

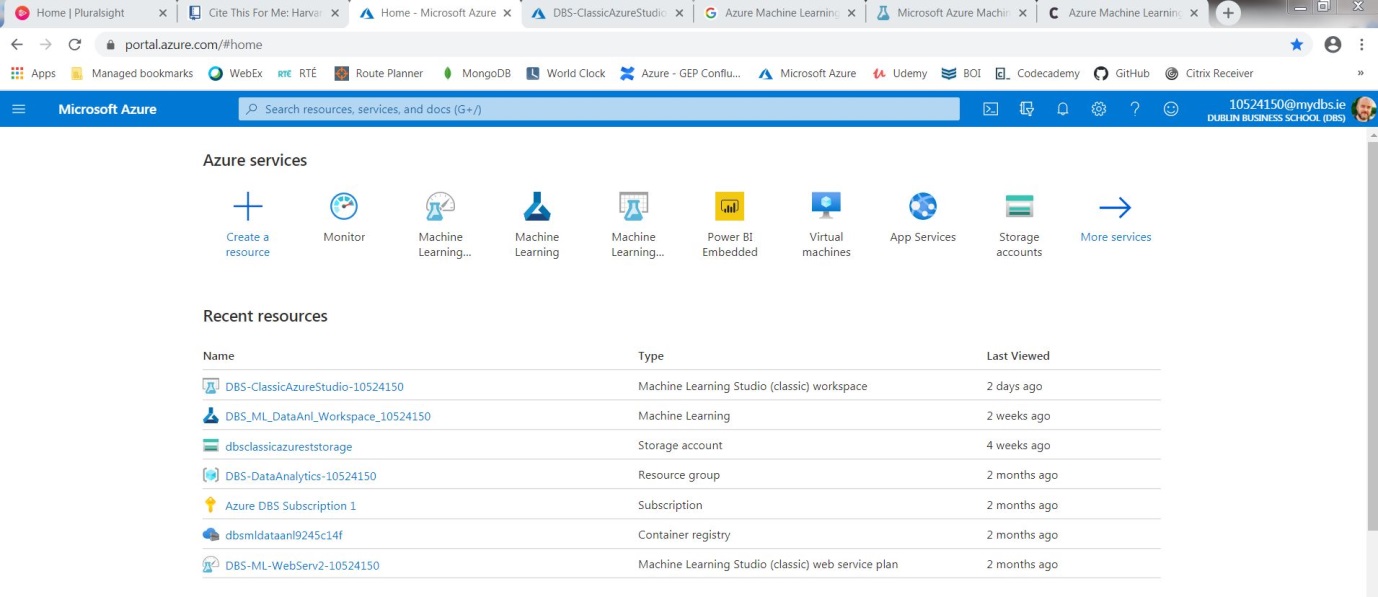
In brief; a description of Azure Workspaces can be found here; <https://docs.microsoft.com/en-us/azure/machine-learning/concept-workspace>

A description of the Azure Machine Learning Studio/Services offering is described here;

<https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-machine-learning-studio>

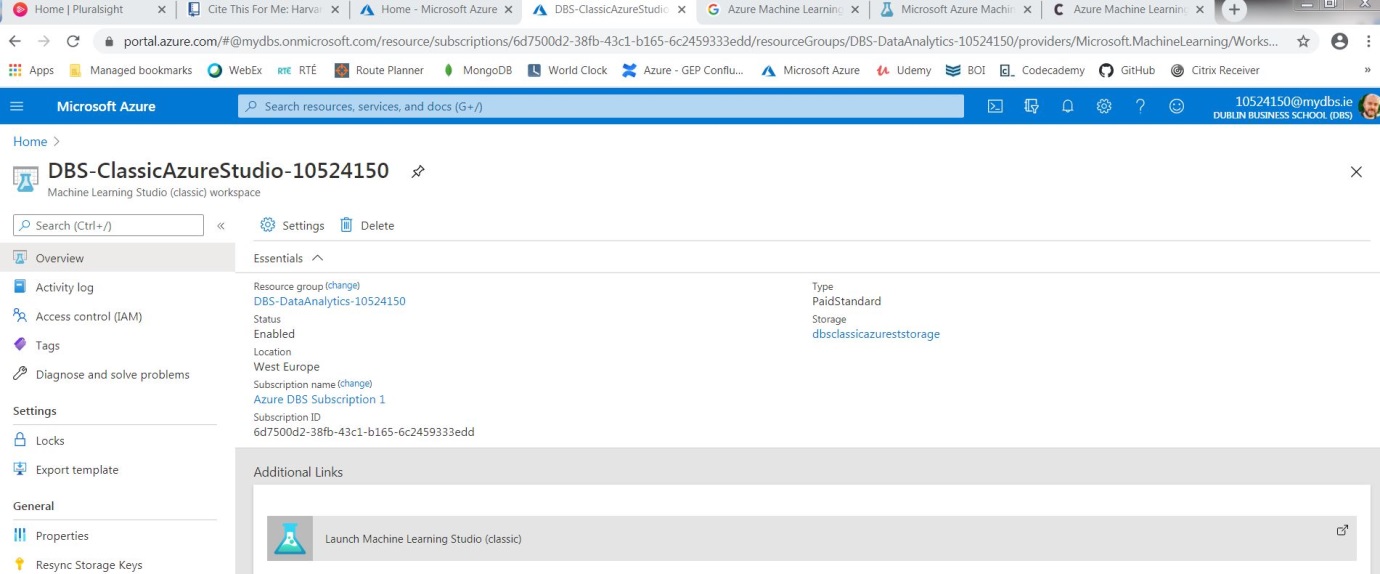
To access the Azure Machine Learning Studio (Classic) where I developed my project the first step is to log onto the Azure Portal, which I set up with my DBS account.

*Figure: Azure Portal (my DBS account)*



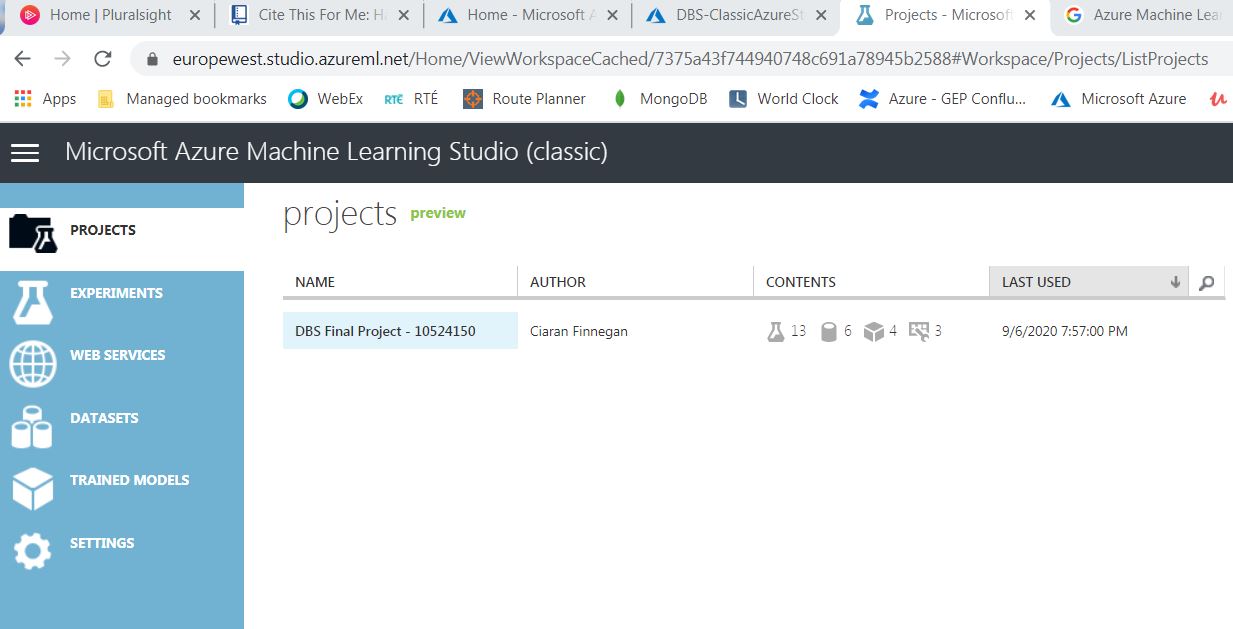
A workspace has been created by me for a Machine Learning Studio (Classic) service.

Figure: Azure ML Studio (Classic) Workspace



Launching the Machine Learning Studio (Classic) services will, after additional user verification, open the ML Studio (Classic) application itself.

Figure: Microsoft Azure Machine Learning Studio (classic)



This ML Studio follows many of the conventions of similar products on the marketplace in terms of organising work under a ‘Project’ structure.

This is a screenshot of my project work and the Project groups together;

* Datasets
* Experiments

The Experiments use the datasets, or outputs of other experiments, to build up the predictive credit card fraud model for this project.

Once ready, my ‘final’ experiment is promoted to a ‘Web Service’ which can then be invoked externally (by my Shiny R application in the case of this project).

The following sections are a sequential analysis of the experiments used in the ‘Project’ to progress through all the steps of the Machine Learning process.

Experiments have been numbered in sequence but the machine learning process has iterated backwards and forwards across the experiments as refinements and alternative options were identified.

**Note: Why use the ‘Classic’ version of the Microsoft Machine Learning Studio?**

In his brief article from 2019 on Codit, Sriram Narayanan***n***, describes the additional features that the more modern Microsoft ML Services offers in comparison to the ‘classic’ studio. Microsoft itself tries to encourage use of the more modern ML ‘Services’ interface.

Working iteratively through the prototype phase of this project, I determined that the ‘classis’ studio was a better option for this delivery for the following reasons;

* **Cost**. The Azure charge for the ‘classic’ studio is very low, and includes the deployment of Web Services / Endpoints. Azure Machine Learning Services is significantly more costly for deploying REST Endpoints on AKS Clusters.
* **Complexity and maturity**. Some of the deployment aspects of Microsoft Azure Machine Learning Services are still in ‘preview’ mode. During Prototype development I had to re-code errors within the Python in certain Jyputer Notebooks when using ML Services examples. I believe that the ‘classic’ option was a more robust platform on which to develop a full ‘end-to-end’ solution.
* **Training**. The Pluralsight courses, to which I had access, had a greater range of training material on ‘classic’ and were an important reference tool for me on this project.

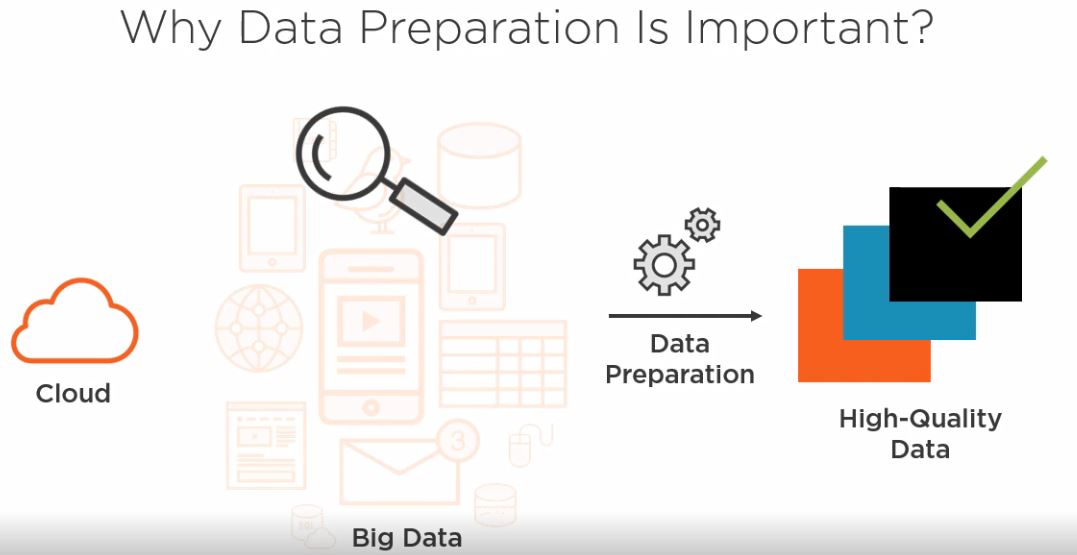
*Figure: Overview of Azure ML Studio (classic) environment for this project – Experiment View*

A screenshot of a computer screen

Description automatically generated

## Credit Card Fraud Dataset – Analysis and Preparation

Figure: *Reproduced Image from Pluralsight*  (Srinivasulu, 2019)



Section 2 of this document described the importance of ‘Feature Engineering’ in then general creation of a credit card fraud predictive model.

To focus specifically on my dataset, feature engineering was important because:

* *My original credit card dataset has 380 columns*. Almost certainly, only a fraction of these columns contains information that will directly influence the accuracy of the final model. It will be necessary to identity those columns that build the most accuracy and performant predictive model for credit card fraud.
* *The dataset is effectively ‘clean’ but still needs to be checked for ‘invalid’ data*. There are no invalid characters in the dataset rows, but missing or useless data needs to be identified, if present.
* *40 columns in the original dataset are non-numeric features and will need some form of re-coding*. Many machine learning algorithms can process non-numeric features, but accuracy is likely to be improved if String features are manipulated before the modelling process begins.

This section of the document details the set up and execution of the following experiments:

* Experiment 1: Data Cleansing
* Experiment 2: Feature Engineering
* Experiment 3: Feature Selection

Exploratory Data Analysis (EDA) is carried out throughout these experiments but the Shiny App UI provides useful graphical descriptions of the dataset. This can be seen in Section 5.1 of this document.

### Experiment 1: Data Cleansing

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged for the data cleansing routines.

*Figure: Experiment 1: Data Cleansing*

A screenshot of a cell phone

Description automatically generated

Appendix 9.3 of this document details the specific steps in this experiment.

The result of this experiment can be summarised as:

* Dataset reduced to 250 columns of potentially ‘useful’ data.
* Top 5% of outlier values in transaction amount ‘clipped’ to reduce distortion in modelling process.
* Generation of an interim dataset for use in Experiment 2.

### Experiment 2: Feature Engineering

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged for the feature engineering routines.

*Figure: Experiment 2: Feature Engineering*

A screenshot of a cell phone

Description automatically generated

Appendix 9.3 of this document details the specific steps in this experiment.

The result of this experiment can be summarised as:

* Conversion of String datatypes to ‘Categorical’ features
* Grouping of Country Code categorical data and numerical encoding of all categorical features.
* Balancing of dataset (via R code routine) to a 50/50 Fraud/Non-Fraud split.
* Identification and removal of a sub-set of highly correlated features.
* Generation of another interim dataset, which will be the input for Experiment 3.

### Experiment 3: Feature Selection

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged for the feature selection routines.

*Figure: Experiment 3: Feature Selection*

A close up of a map

Description automatically generated

Appendix 9.3 of this document details the specific steps in this experiment.

The result of this experiment can be summarised as:

* Taking the output of the feature engineering steps in Experiment 1 + 2 and generating a predictive fraud model.
* Obtaining a list of features scored in order of importance to the predictive model. The ‘Permutation Feature Importance’ module produces this output.

This experiment was run multiple times with various modelling algorithms, based on comparisons seen in later experiments. The ‘Two-Class Logistic Regression’ algorithm provided the best performing and accurate model and was, hence, used to determine the final list of parameters selected for the model.

This choice of features has a direct impact on the feature set captured in the Shiny App UI and passed to the Rest Endpoint for the predictive model.

## Credit Card Fraud – Building the Azure Model

After a series of iterations backwards and forwards through the experiment sequences, I believed that I now had a refined credit card dataset with which I could run a final batch of modelling experiments.

*Figure: Representation of ML Modelling Process Reproduced from Edureka (Lateef, 2020)*

A close up of a map

Description automatically generated

The pattern of operations followed the illustration above, but my primary objectives were:

* Determine which classification algorithm, which is available for use in Azure Machine Learning Studio (classic), would be most effective for generating a predictive fraud model based on my credit card transaction dataset. Criteria for algorithm selection would be:
  + Accuracy Score
  + Recall – how well actual Fraud is detected
  + Performance
* Demonstrate the impact on fraud prediction model accuracy, and other metrics, introduced by the following modelling actions:
  + Feature Engineering
  + Cross Validation
  + Hyperparameter tuning

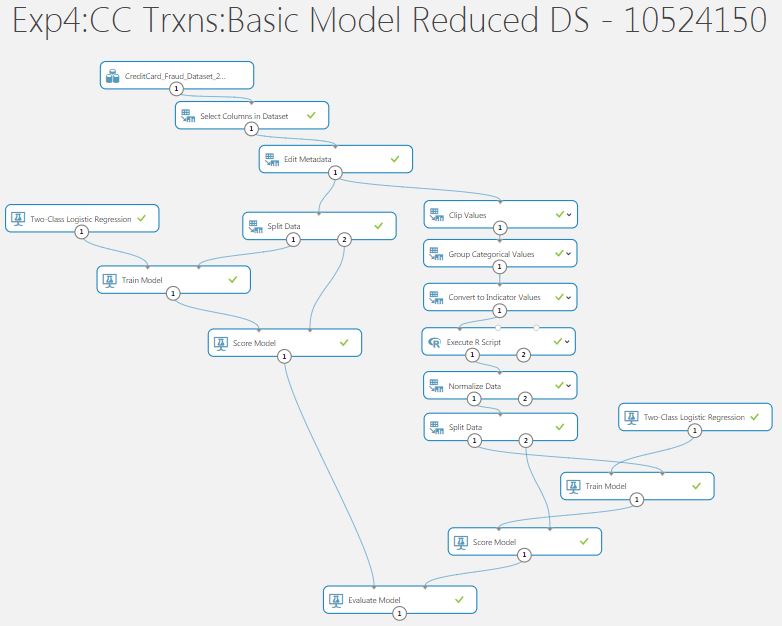
This section of the document details the set-up and execution of the following experiments:

* Experiment 4: Basic Model Evaluation with Feature Engineering
* Experiment 5: Model Evaluation using Cross Validation and Hyperparameter tuning
* Experiment 6 + 7: Comparison of Multiple Classification Algorithms

### Experiment 4: Basic Model Evaluation with Feature Engineering

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged to assess the benefits of Feature Engineering.

*Figure: Experiment 4: Feature Engineering and Model Evaluation*



Appendix 9.3 of this document details the specific steps in the left hand side (LHS) and right hand side (RHS) of this experiment as they largely replicate the work in Experiments 2 and 3.

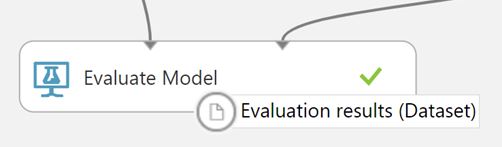
The result of this experiment can be summarised as:

* A demonstration of the impact of feature engineering on model accuracy and other metrics.
* Possible trade-offs that might be acceptable in the modelling process.

Again, ‘Two-Class Logistic Regression’ is used because of evaluation results in later experiments feeding back into this ‘final’ version of Experiment 4.

*Model Evaluations*

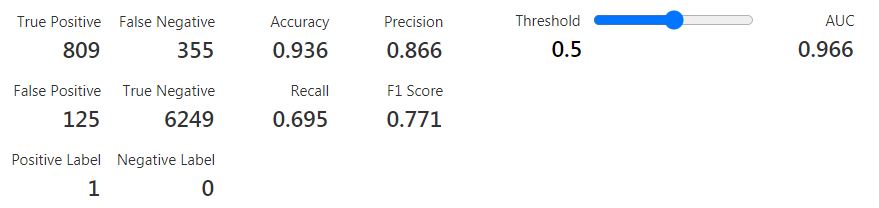
The ‘Evaluate Model’ module provides takes two inputs and provides the key scoring metrics on comparative models as an output.



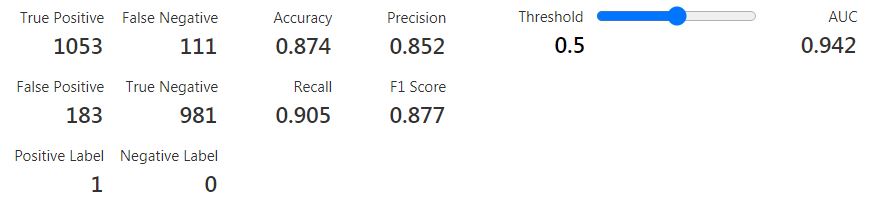
In Experiment 4:

* The ‘*Scored dataset*’ was the model generated without Feature Engineering, except for the conversion of String features into Categorical features.
* The ‘*Score dataset to compare*’ was the model generated with the Feature Engineering routines in Experiments 1, 2, and 3.

The ‘*Scored dataset*’ produced the following scores:



The ‘*Scored dataset to compare*’ produced the following scores:



*Model Score Assessments*

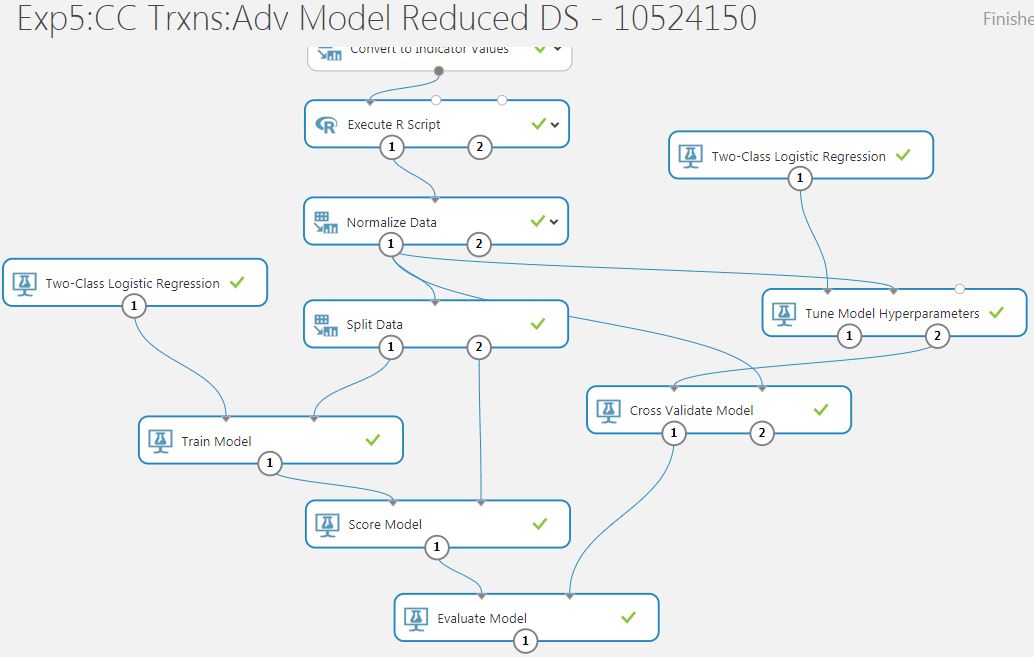
Feature Engineering does not improve the overall accuracy of my credit card predictive model for fraud, but it is much better at detecting actual fraud cases (higher recall value).

### Experiment 5: Model Evaluation with Cross Validation/Hyperparameter Tuning

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged to assess the benefits of using Cross Validation and hyperparameter tuning.

*Figure: Experiment 5: Cross Validation and Hyperparameter Tuning*

*Note:- This image has been deliberate truncated to focus on the modules after Feature Engineering.*



Appendix 9.3 of this document details the specific configurations of the ‘Tune Model Hyperparameter’ and ‘Cross Validate Model’ modules.

The result of this experiment can be summarised as:

* Experiment 4 conducted a straightforward Test/Train split of the dataset for modelling. Can we determine if Cross Validation will improve the reliability of my predictive model for credit card fraud detection?
* Azure Machine Learning Studio (classic) allows for an automated process to tune the hyperparameter values on an algorithm. Does this also contribute to better fraud prediction for my dataset?

Again, ‘Two-Class Logistic Regression’ is used because of evaluation results in later experiments feeding back into this ‘final’ version of Experiment 5.

*Model Evaluations*

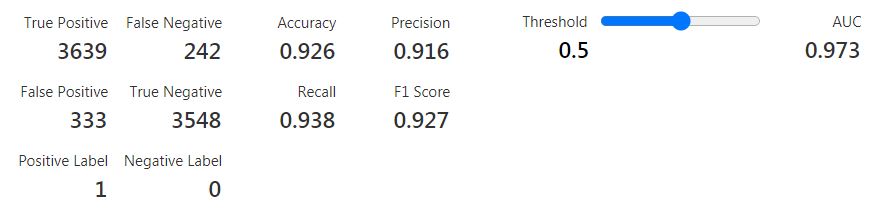
As before, the ‘Evaluate Model’ module provides takes two inputs and provides the key scoring metrics on comparative models as an output.

In Experiment 5:

* The ‘*Scored dataset*’ was the model generated with Feature Engineering in Experiment 4.
* The ‘*Score dataset to compare*’ was the model generated using Cross Validation on the dataset and tuned hyperparameters for the Two-Class Logistic Regression algorithm.

The ‘*Scored dataset’* is unchanged from Experiment 4.

The ‘*Scored dataset to compare*’ produced the following scores:



*Model Score Assessments*

Using Cross Validation and hyperparameter tuning in Experiment 5 has produced a model that scores almost as well in ‘Accuracy’ as the LHS model Experiment 4 (0.936 vs 0.926).

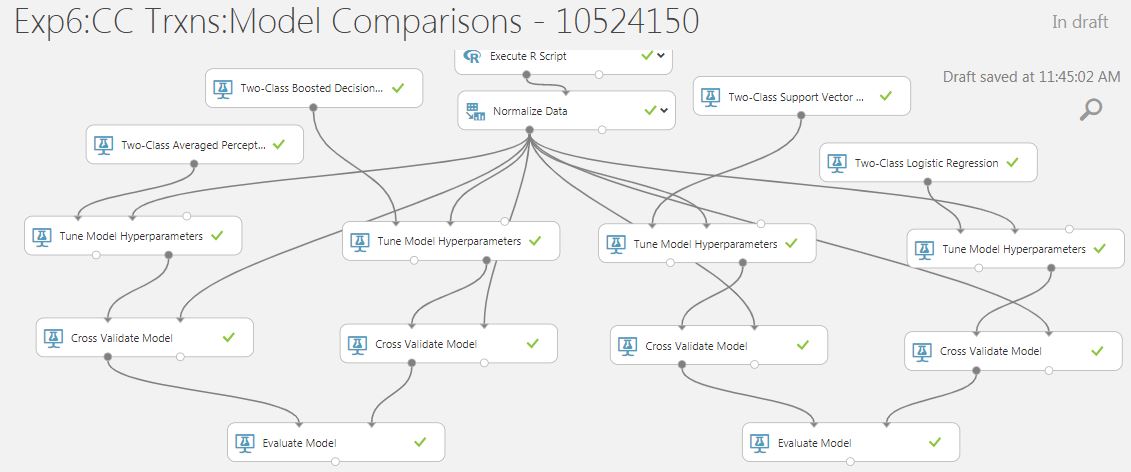
However, the ‘Recall’ score in Experiment 5 is higher again (0.938) and is thus even better at finding Fraud that either of the models in Experiment 5.

### Experiment 6: Comparison of Multiple Classification Algorithms (1)

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged to assess the performance of multiple Classification algorithms.

*Figure: Experiment 6: Comparing Classification Algorithms*

*Note:- This image has been deliberate truncated to focus on the modules after Feature Engineering.*



Based on the results from Experiment 5, Cross Validation and hyperparameter tuning will be applied to all models built in further experiments to create my credit card predictive model for fraud detection.

The result of this experiment can be summarised as:

* Compare results of four similar ‘Two-Class’ Classification algorithms when creating a predictive model based on my credit card fraud dataset. The algorithms being compared in this experiment are:
  + Two-Class Averaged Perceptron.
  + Two-Class Boosted Decision Tree.
  + Two-Class Support Vector Machine.
  + Two-Class Logistic Regression.

The selection of classification algorithms in the Azure Machine Learning Studio (classic) is limited to nine options, of which I choose eight. The other classification algorithms specialise in multi-class problems.

*Model Evaluations and Assessments*

Appendix 9.3 of this document provides a breakdown of all Experiment 6 and 7 evaluation scores for each model.

**‘Two-Class Logistic Regression’ performs best, based on a combination of ‘Accuracy’ and ‘Recall’.**

Other observations on the algorithm performances were (based on a 25K row dataset with 39 features):

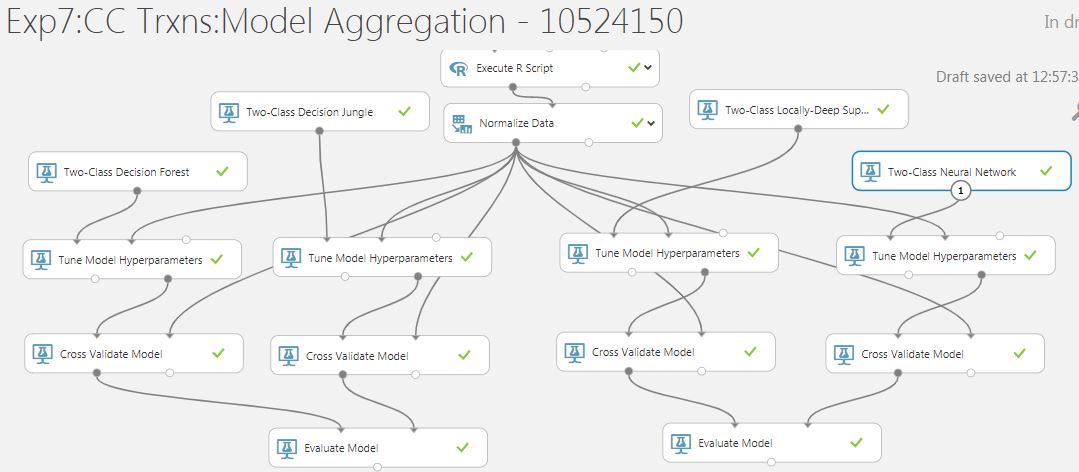
* The Two-Class Averaged Perceptron algorithm was the quickest to run (< 1 minute) and complete. The Microsoft documentation describes this as a simplified version of a neural network. It is sometimes favoured when the goal is speed over accuracy. (Microsoft, 2019).
* The Two-Class Boosted Decision Tree took the longest to run and complete. Hyperparameter tuning alone took 10+ minutes, and the model was not available for scoring for nearly 20 minutes. The Azure Machine Learning Studio (classic) contained a tutorial recommending this algorithm for client credit risk solution, but performance with my dataset was a concern. (Normalization was probably a redundant step in this modelling process but was left in place for simplicity.)
* The Two-Class Support Vector algorithm took 5+ minutes to complete the modelling process. (The second longest). Microsoft documentation recommends this for simpler datasets where the aim is, again, speed over accuracy. Results were good but performance was slow.
* Two-Class Logistic Regression was dependent on conversion of non-numeric features but performed the best overall.

### Experiment 7: Comparison of Multiple Classification Algorithms (2)

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged to assess the performance of further multiple Classification algorithms.

*Figure: Experiment 7: Comparing Classification Algorithms – Pt2*

*Note:- This image has been deliberate truncated to focus on the modules after Feature Engineering.*



The result of this experiment can be summarised as:

* Compare results of four similar ‘Two-Class’ Classification algorithms when creating a predictive model based on my credit card fraud dataset. (). The algorithms being compared in this experiment are:
  + Two-Class Decision Forest.
  + Two-Class Decision Jungle.
  + Two-Class Locally-Deep Support Vector Machine.
  + Two-Class Neural Network.

These are possibly more complex algorithms with greater processing overhead and are included in the project to compare with the group of algorithms in Experiment 6.

*Model Evaluations and Assessments*

Appendix 9.3 of this document provides a breakdown of all Experiment 6 and 7 evaluation scores for each model.

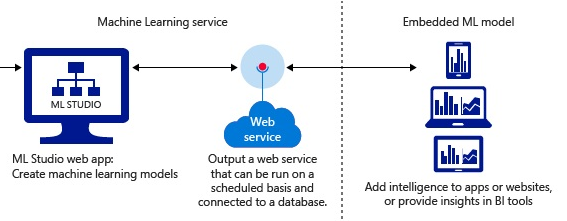
None of the Experiment 7 algorithms generated superior results, in terms of ‘Accuracy’ and ‘Recall’ when compared to the Two-Class Logistic Regression based model.

## Credit Card Fraud – Deploying the Azure Model

My iterations through the experiments to evaluate the best algorithm, including the optimum training process, provided me a training model that I now wanted to deploy into production.

This would allow me to host the model in Azure and invoke that model to display fraud predictions on ‘new’ credit card transactions.

*Figure: Reproduced from MicroStrategy Community*(Sonobe, 2017)



My objectives, at this stage of the project lifecycle, were to:

* Prepare and validate a ‘final’ model based on my refined feature engineering routines and training process, using my chosen classification algorithm.
* Create a ‘Predictive’ version of the trained model, in preparation for the set-up or a Web Service to be hosted in Azure. Deploy this credit card predictive fraud model as a Web Service hosted in Azure and test the deployment with sample data.
* Update the Shiny App UI code with R code that invokes the API to return a real time predictive score on the likelihood of Fraud for a given new credit card transaction, selected by the user through the project UI. (See Section 5 of this document for details on the code routines to extract key data elements from ‘new’ card transactions and pass them to the API for the fraud model).

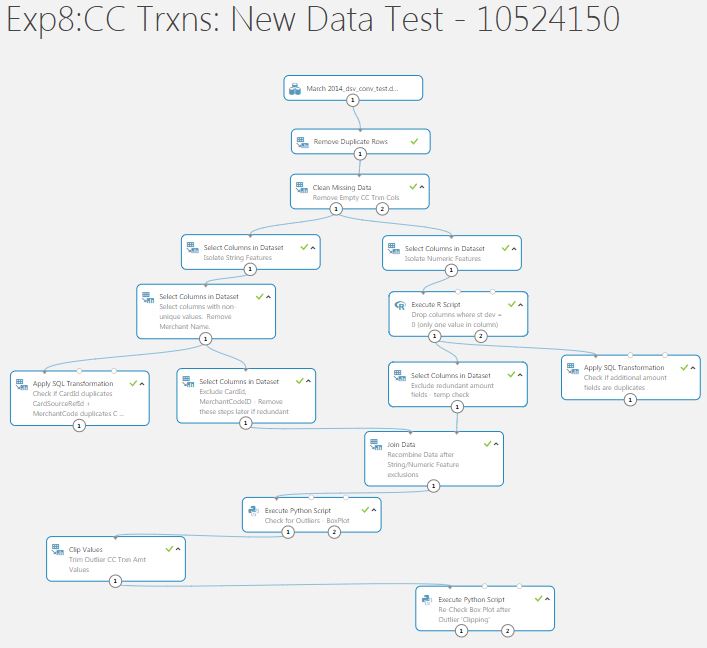
This section of the document details the set-up and execution of the following experiments/actions:

* Experiment 8: Repeat of initial Feature Engineering routines with larger dataset.
* Experiment 9: Creation of ‘Predictive’ model.
* Deployment and validation of Web Service for trained model.

### Experiment 8: Feature Engineering on Larger Dataset

The figure below illustrates how the Azure Machine Learning Studio (classic) modules were arranged to assess process initial Feature Engineering routines on the larger credit card transaction dataset.

*Figure: Experiment 8: Feature Engineering on Larger Dataset*

**

Experiment 8 is a re-execution of Experiment 1 but on the larger 25K dataset.

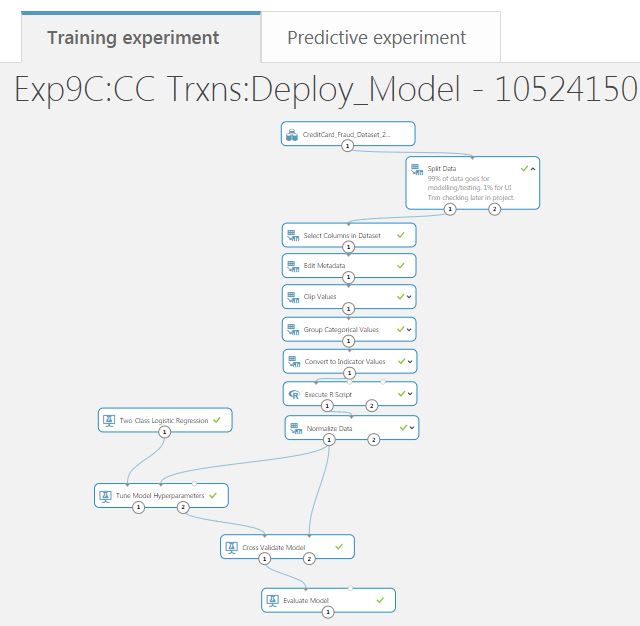
The larger dataset is being introduced at this point in the project to provide a greater volume of data for the training process, and thus ideally increase the reliability of the predictive fraud model.

### Experiment 9: Creation of Predictive Fraud Model for Deployment

Experiment 9 is drawn from the results and conclusions from earlier experiments.

The ‘Training Experiment’ in the illustration below represents the ‘final’ model creation approach.

*Figure: Experiment 9 : Training Model*



(The ‘9C’ numbering convention is the result of a number of iterations on this experiment).

The purpose of this Experiment is to create a training model which will then be converted into a ‘Predictive’ model.

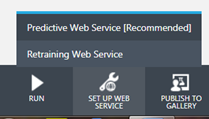
The ‘Predictive’ model is the basis for the deployment of a Web Service to allow external access (from my Shiny R application) to the scoring model for credit card fraud.

The experiment above is under the ‘Training’ tab.

*Generating a Predictive Experiment*

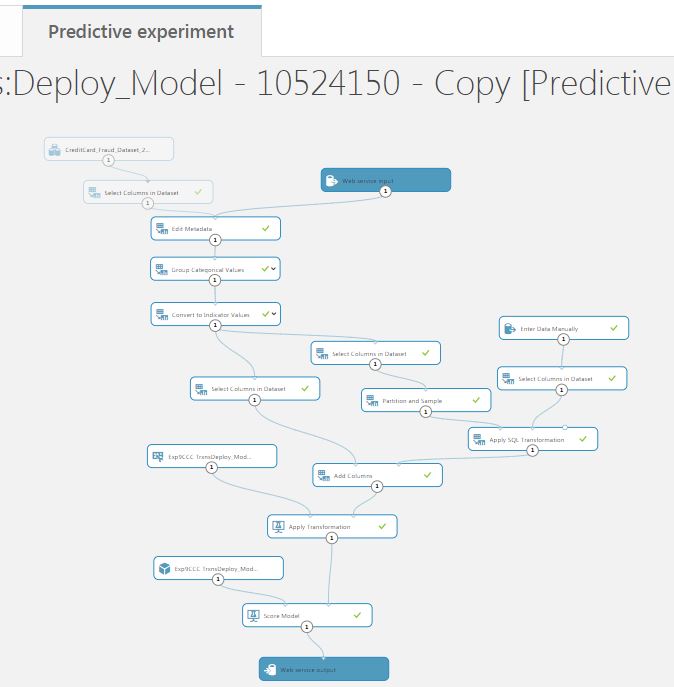
The Azure Machine Learning Studio (classic) provides an option for any experiment with a trained model to be deployed as a Web Service.

*Figure: Option to generate Predictive experiment*



This creates a ‘stripped down’ version of the Training experiment called the ‘Predictive experiment’.

*Figure: Experiment 9: Predictive experiment*

**

*Web Service Inputs / Outputs*

As some of the remaining modules will be redundant for the real time ‘one-by-one’ scoring of transactions, which is a key requirement of my credit card fraud prediction project, I have removed other elements of the ‘Training experiment’ that were brought across.

Key features of the Predictive experiment, as show in the illustration, are:

* Azure Machine Learning Studio (classic) introduces Web Service input and output modules. These determine the interface points to the model in deployment.
* I moved the Web Service input to a point after the Feature Selection module. This is done so that the API code in my Shiny R application will only need to pass the sub-set of features directly required by the module, and not the much larger (post Feature Engineering) dataset.
* I have created additional modules to generate a manual one-hot encoding. This is required because tests failed during verification of the deployed model when transactions were being processed ‘one-by-one’. Single credit card transactions would not generate the additional non-numeric features created during the modelling process with the larger dataset.

One validated the Predictive experiment can be deployed as a Web Services, hosted within Azure.

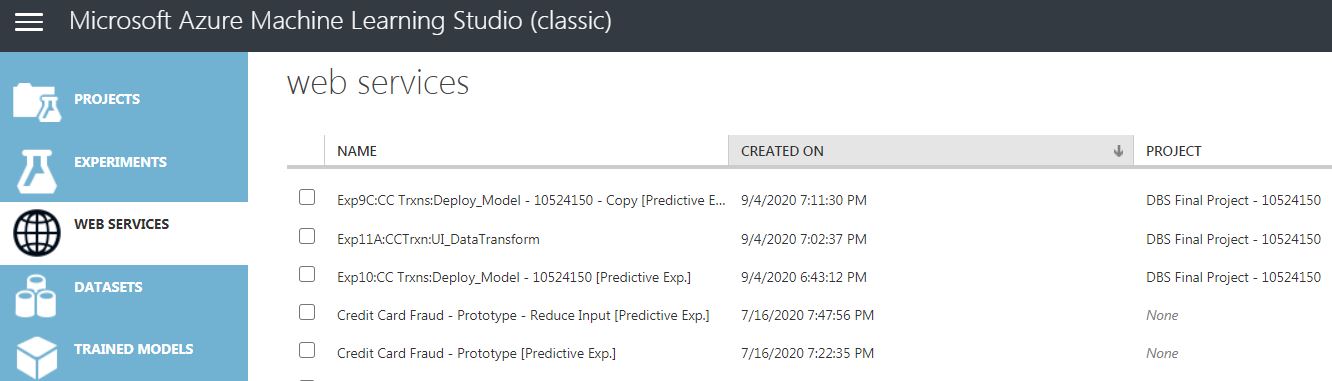
*Figure: Option to Deploy Web Service*



### Deployment and Validation of Web Service for Predictive Fraud Model

Azure Machine Learning Studio (classic) maintains a list of generated Web Services, which can be accessed through the Studio interface.

*Figure: Web Services List*

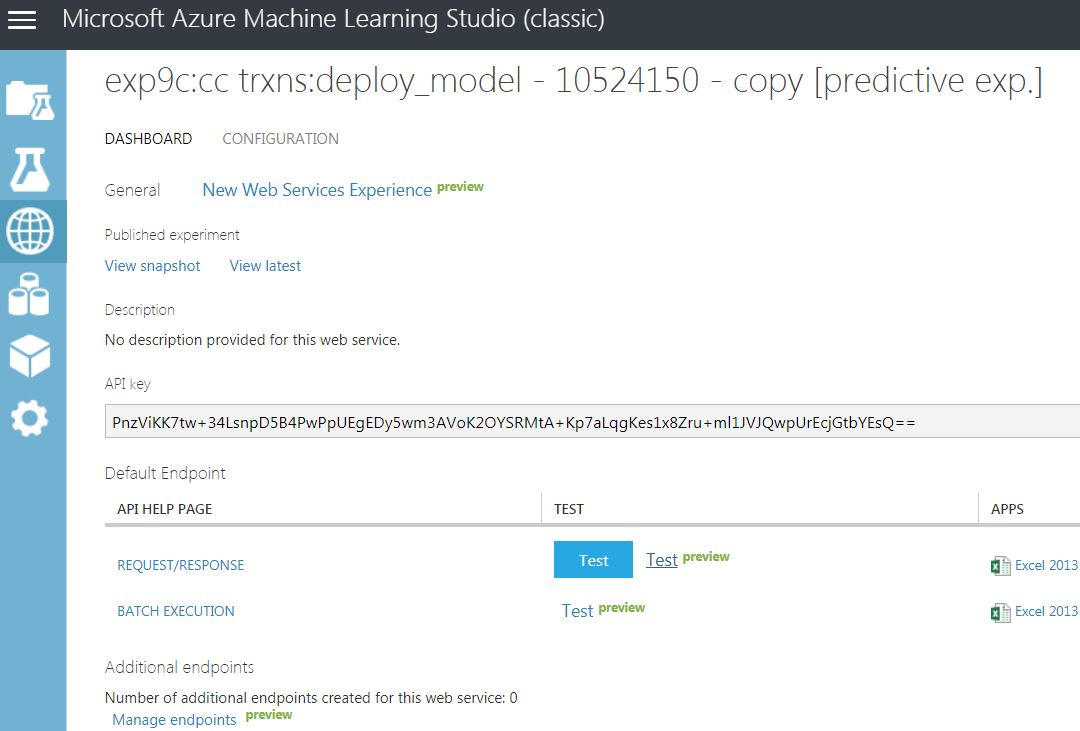


The Web Services that I have generated in my project as part of ongoing research, for the Interim Prototype, and for the final predictive credit card fraud model can be seen in this illustration above.

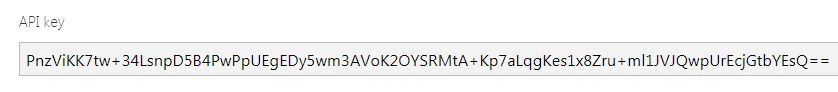
*How to validate the Web Service through Azure?*

The generation process for a Web Services, if successful, brings the user to a dashboard screen. The illustration below shows the dashboard screen for my final production model for credit card fraud prediction.

*Figure: Web Services Dashboard*



Section 5 of this document will explain more about how this Web Service is consumed but a key element on this page is the API key through which the model can be invoked externally.

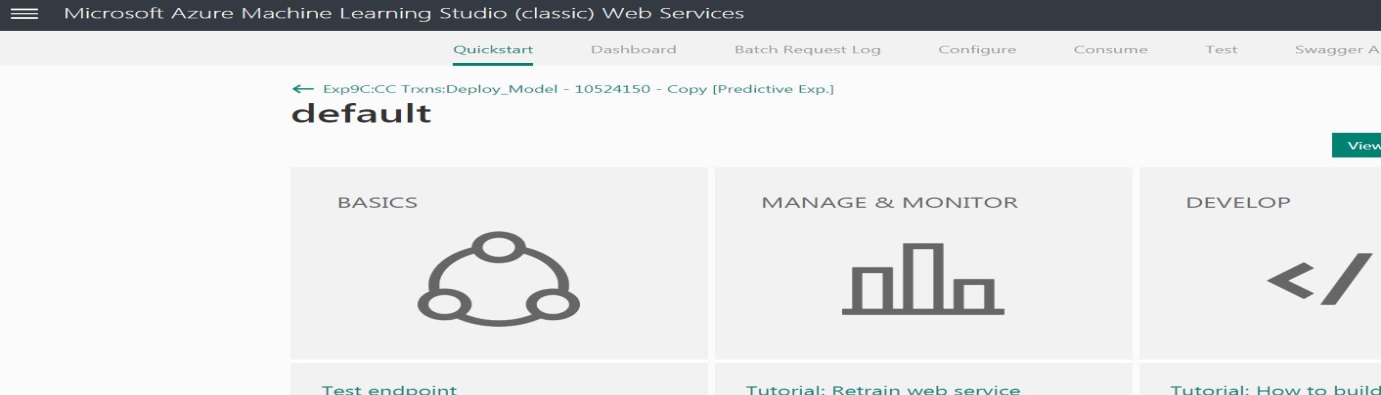


Before I began writing R code to access the API for the fraud model, I needed to verify that the Web Service was working as expected and returning a score for predicting fraud on my credit card transaction.

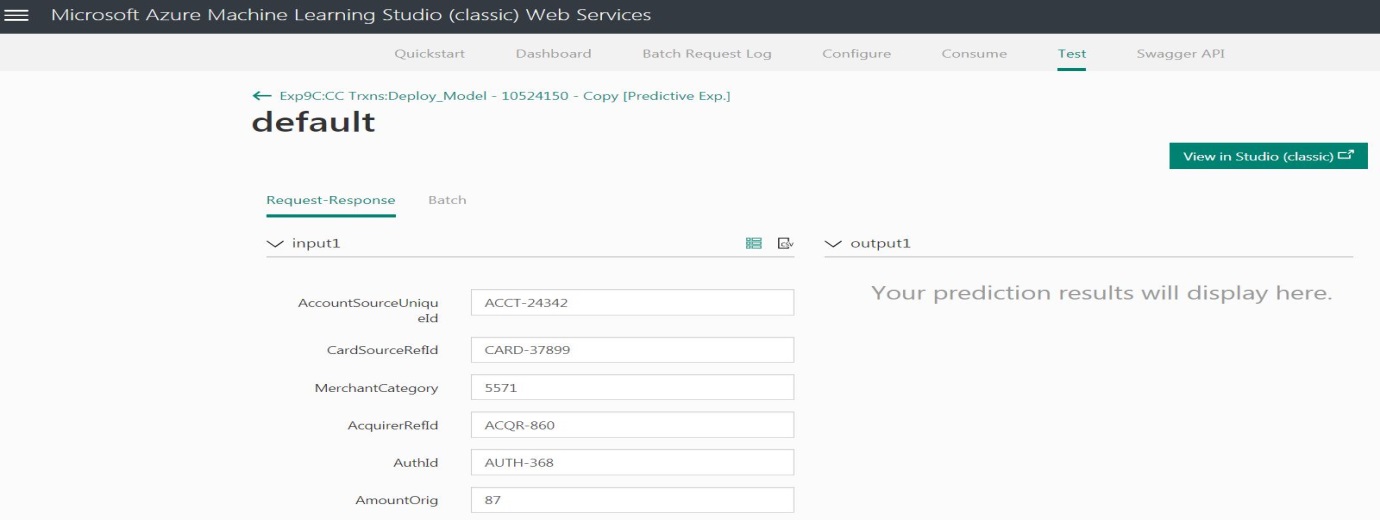
Azure Machine Learning Studio provides a separate Web Services portal to test and manage these hosted endpoints.

Using this portal, shown in the illustrations below, I was able to validate my credit card fraud predictive model was working correctly.

*Figure: Azure ML Studio Web Services Portal – Main Screen*



*Figure: Azure ML Studio Web Services Portal – Test Screen*



Test results return a ‘Score Label’ – ‘1’ for Fraud, ‘0’ for Non-Fraud. A Scored Probability value is also returned, which is a number between 0 and 1 (> 0.5 = Fraud, <0.5 = Non-Fraud).

# Project Implementation (2) – Shiny R Dashboard UI

## Data Visualisations in a Shiny Dashboard

The prototype for this project application..

The..

<images>

## Credit Card Fraud – UI to Check Fraud Predictions

The..

<images>

## Shiny UI – Hosted Application

The..

<images>

# Testing and Results

## User Story ‘Demos’ – Test Results and ‘Feedback’



### User Story 4: Initial Data Modelling – Review and Evaluation

Goal: *Build a basic credit card fraud predictive model in Azure ML Studio (classic) based on a small subset of transactions dataset*.

Assessment of robustness of code and functionality delivered:

1. Goal Achieved – August 1st 2020. User Stories 1 – 3 provided enough research and background to set up Azure ML workspace for ML Studio (classic).

*Figure: User Story 4 demonstration*

A screenshot of a cell phone

Description automatically generated

1. Model generated with manual selection of features and elementary. Tests with ‘Evaluate Model’ module displayed Accuracy results of ~82%. ‘Recall’ value extremely poor but model acceptable for prototype.

*Figure: User Story 4 Test Model Results*

*A screenshot of a cell phone

Description automatically generated*

### User Story 5: Basic Shiny App – Review and Evaluation

Goal: *Build a basic Shiny R dashboard app that displays basic EDA of my credit card dataset and has a placeholder screen for fraud detection interface.*

Assessment of robustness of code and functionality delivered:

1. Goal Achieved – August 7th, 2020.

*Figure: User Story 5 demonstration*

*A screenshot of a cell phone

Description automatically generated*

1. Quick Turnaround from User Story 1. Basic Shiny Dashboard App running without error from within RStudio environment.

### User Story 6: Integrated Prototype – Review and Evaluation

Goal: *Add R code to R Shiny Dashboard to invoke basic card fraud model with fixed data inputs. Host working Shiny App online.*

Assessment of robustness of code and functionality delivered:

1. Goal Achieved – August 14th, 2020. This working prototype was released online with a basic user guide as part of the Interim Report for the project.

*Figure: User Story 6 demonstration*

A screenshot of a cell phone

Description automatically generated

### User Story 7: Enhanced Modelling – Review and Evaluation

Goal: *Refine credit card model with full ML workflow processes. Enhance UI to select ad-hoc credit card transactions.*

Assessment of robustness of code and functionality delivered:

1. Goal Partially Achieved – September 5th, 2020.
2. Full end-to-end ML workflow applied to create a production ready model for credit card fraud prediction. Tested and validated in the Azure ML Studio Web Services portal.

*Figure: Web Services Portal Testing of ‘final’ predictive model.*

A screenshot of a cell phone

Description automatically generated

1. Shiny App UI only partially updated. Complexity of rebuilding model left no time to complete this section of the User Story. The UI is reading in new fixed files but there is no option to select a transaction file at random by the user. Carried over to User Story 8.

### User Story 8: Enhanced UI – Review and Evaluation

Goal: ***(Revised)*** *Redeploy new credit card fraud model in Azure. Update code in Shiny R Dashboard to:*

* *Invoke new API*
* *Allow for ad-hoc selection of ‘new’ transactions to submit to predictive fraud model*
* *Display improved data visualisation graphs on UI based on credit card dataset*

Assessment of robustness of code and functionality delivered:

1. Goal Achieved – September 17th, 2020.
2. The user can select from multiple files and submit any given transaction for fraud assessment.
3. The Dashboard tab containing data visualizations of the original credit card dataset has been enhanced with additional graphs.

A screenshot of a social media post

Description automatically generated

### User Story 9: Presentation Preparation – Review and Evaluation

Goal: *Refine UI in preparation for Final project demonstration.*

Assessment of robustness of code and functionality delivered:

1. Goal Achieved – September 25th, 2020.
2. Code complete, including all code refactoring and commenting.
3. All tests completed and passed.
4. Documentation completed and proofread.
5. Project presentation completed.
6. Project submitted.

A screenshot of a computer

Description automatically generated

## Final Project Assessment: A Critical Evaluation

The project is intended to demonstrate and end-to-end solution for credit card fraud detection, using established and comprehensive Machine Learning techniques.

*An important section in this chapter is the critical evaluation of the final project, where the student demonstrates the ability to critically evaluate the work done, the shortcomings in the project and so on. Objectivity is important when writing this chapter.*

Taking a checkpoint at the end of the project, I feel I have been largely successful in the goals I set out for myself in this project. However, not all aspects of the implementation have gone as planned, and there are some inherent limitations with the approach I took.

On balance, these are my primary evaluations and observations on the project:

1. In section 2.2 of the Interim Report, I describe how a ‘real-world’ fraud detection system would almost certainly process credit card transactions in a large-scale batch mode. Those transaction marked as possible fraud would then be sent, usually though an internal company workflow system, into some kind of ‘Case Management’ system for a Fraud Investigator to review. Despite this, I feel the experience of building a real time one-by-one fraud detection interface has been an excellent learning experience for me, and a great academic challenge.
2. The Microsoft Azure Machine Learning Studio (classic) was interesting tool to learn to use, and to build and deploy the predictive mode for this project. For the most part I used the visual designer aspect of the tool. There were elements of embedded R and Python code it the experiments for Feature Engineering, but these were for relatively atomic tasks. This was a challenging project to implement but my next iteration of this project would raise the academic stakes by writing much more code in Python, within frameworks such as Jupyter Notebooks.
3. The Shiny R dashboard is reasonably aesthetically pleasing, thanks largely to the Semantic libraries used within the Shiny R development. However, there is room for improvement. The visual graphs on the first tab are only static representations of the dataset and do not take advantage of any of the interactive possibilities in Shiny. The predictive fraud interface could also use a little ‘polish’ when new credit card transactions are chosen but I found this difficult to implement.

## Project Plan 2020: Final Status – 25th September 2020

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

A screenshot of a video game

Description automatically generated

# Project Location and User Guide

## 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 (Final Project)

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

*Figure: Final Project*

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…

# Project Conclusions

## Where Project Goals Achieved?

**Yes**. Looking at the architecture diagram in Section 3.3 of the Interim Report, and reproduced in Section 3.2 of this document, I feel I built the application that I set out to create.

The Machine Learning process in the Microsoft Azure Machine Learning Studio (classic) platform built and deployed a model that performed with very satisfactory results.

Invoking the Web Services for the model through the Shiny R Dashboard encountered some challenges but I was pleased that the interface met the requirements I set out at the start of the project.

The overarching goal of using this project to cement the knowledge I learned throughout my Data Analytics course in DBS in 2019/2020 was most certainly achieved (IMHO).

## Future Design/Deployment Considerations

*Where did I deviate from the original design requirements, as documented in the Interim report?*

* **Dataset size.** The prototype version of the project application, which was available in tandem with the submission of the Interim Report, worked off a subset of 2.5K transactions. This was more than sufficient for the early phases of development. The final project was to use a larger dataset of 100K+ rows but issues with data formats and file type incompatibility meant that I had to settle for a final dataset size of 25K. This was still 10x times greater than the prototype, and produced an accurate/reliable model, but I would have preferred to work with more information.
* **Modelling Platform.** Although not necessarily a deviation from the original design, I had hoped to look at training and deployment options in the more up to date Azure ML ‘Services’ option for the final version of the project. However, cost and complexity meant that I remained working (successfully) within the Azure ML Studio ‘classic’ version.

What were the learning experiences? Where will the lessons of this project lead for me?

* **R**. Building the UI reinforced my R development skills, learned during my time in DBS.
* **Data Mining**. The data manipulation and modelling techniques required for this project reinforced for me the process and benefits of the Machine Learning workflow process.
* **Career options**. I work in a company that is making the steady evolution from rules-based applications for financial crime prevention to Machine Learning technologies. I make not be able to class myself as an expert in the field (yet) but this project, and the overall DBS course experience, has enabled me to understand the vocabulary of the ML domain. My intention would be to exploit this knowledge and re-orient my career toward an ML Product development path.

*What would be the suggestions for an evolution of this project with further development?*

* **More data**. Although it is in a DSV file format that is not widely supported, and contains 1600 columns of which the vast majority are redundant, I have access to a larger dataset than can provide 280K credit card transaction records. Working with that larger dataset on a new platform (such as AWS Sage Maker or Google Colaboratory) would be an interesting new challenge.
* **More modern ML development platform**. Even if future development chose to remain within the Microsoft Azure platform the classic studio is likely to be deprecated in the near future. Microsoft recommend users move to the newer Machine Learning Services platform, which integrates more seamlessly with technologies such as Jupyter Notebooks and has access to a greater range of Machine Learning algorithms.

# 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*

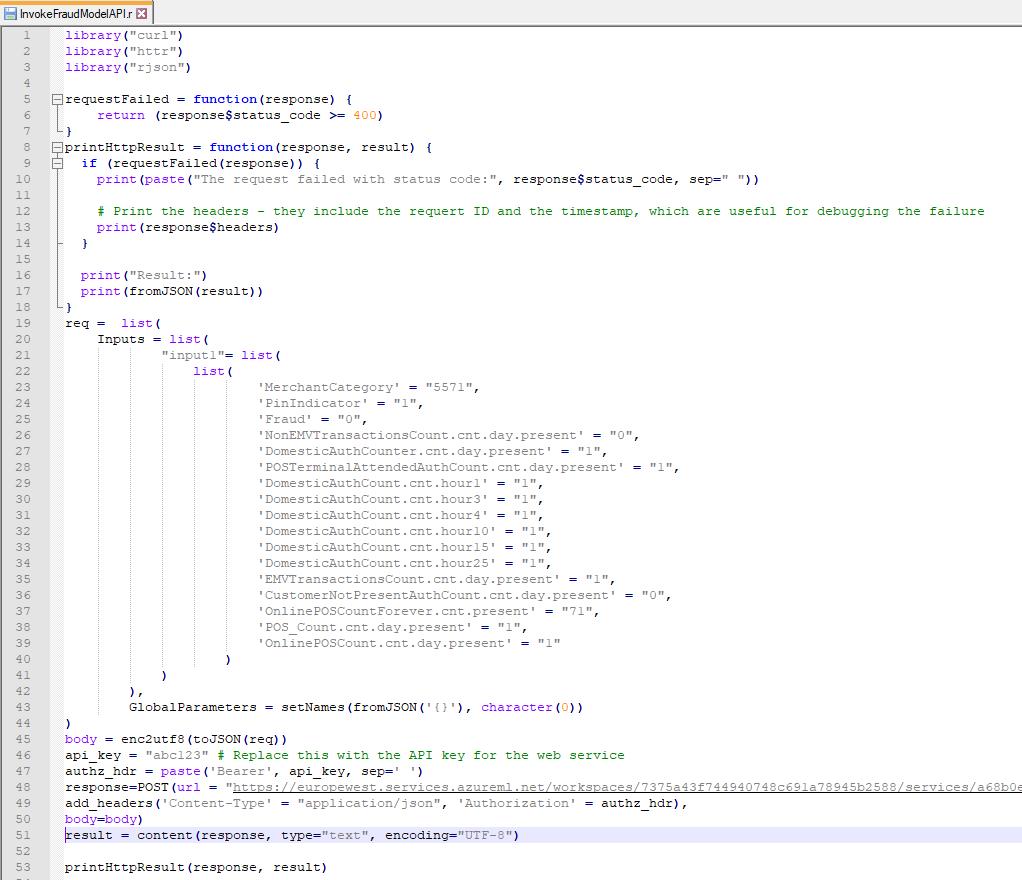
A screenshot of a social media post

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



### Diagram: The RStudio Cloud Environment

A screenshot of a social media post

Description automatically generated

### The Shiny UI Code – Data Visualisations

The..

### The Shiny UI Code – Transaction Fraud Detection

The..

### The R Code Parsing Data and Invoking UI

The..

## Azure Machine Learning Classic Studio Experiments



### Experiment 1: Breakdown

*Breakdown of Experiment*

*Exp1: Step 1. Remove duplicate rows. Remove columns with missing data*

A screenshot of a cell phone

Description automatically generated

Columns with missing data were seen to have a lot of empty cells. Removal was the best/most straightforward option.

The original dataset started with **380** columns. This transformation reduced the dataset to **362** columns.

A picture containing knife, table

Description automatically generated

*Exp1: Step 2. Spilt Numeric/Non-Numeric Features*

A screenshot of a cell phone

Description automatically generated

Split numeric/non-numeric features for subsequent processing.

*Exp1: Step 3. Remove String Columns with Duplicate Data*

A screenshot of a cell phone

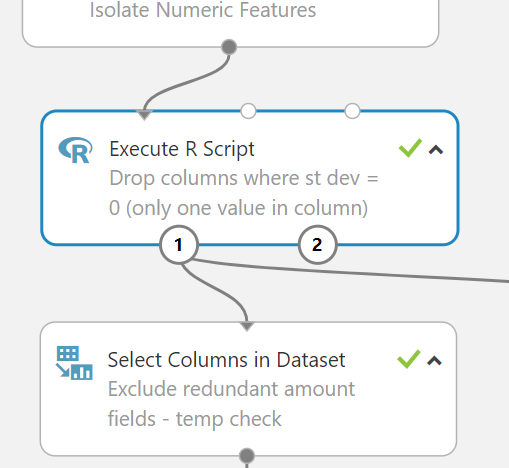
Description automatically generated

A screenshot of a cell phone

Description automatically generated

Use SQL output values to check for columns with duplicate data. This requires a number of iterations and feeds into next ‘Select Columns’ module.

*Exp1: Step 4. Remove Numeric Columns with Only One Value*



A screenshot of a social media post

Description automatically generated

Use embedded R code module to isolate columns that only contain one value. feeds into next ‘Select Columns’ module.

*Exp1: Step 5. Recombine Non-Numeric/Numeric Features*

*A screenshot of a cell phone

Description automatically generated*

*Exp1: Step 6. Check for and Remove Outliers*

*A screenshot of a cell phone

Description automatically generated*

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

An embedded Python routine checks for outliers on cc transaction amounts and outputs a Box Plot representation, which is heavily skewed by transaction amount outliers.

<images>

A screenshot of a cell phone

Description automatically generated

The ‘Clip Values’ module caps the top 5% of amount values to reduce distortion that outliers on transaction amount would introduce the predictive credit card model.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Running another Python routine shows how the impact of outliers is being reduced in the dataset in terms of the transaction amounts for each credit card row.

### Experiment 2: Breakdown

*Breakdown of Experiment*

*Exp2: Step 1. Take Input from Experiment 1*

*A screenshot of a cell phone

Description automatically generated*

*Exp2: Step 2. Convert Strings to Categorical Values / Encode.*

*A screenshot of a cell phone

Description automatically generated*

*Exp2: Step 3. Balance the Dataset 50/50.A picture containing clock

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*

This embedded R code routine takes all the Fraud data and extracts a random sample of the same size from the larger sub-set of non-Fraud transactions. The resultant combined dataset is output to the next module.

*Exp2: Step 4. Normalize Data*

*A screenshot of a cell phone

Description automatically generated*

*Exp2: Step 5. Check Numeric Columns for high Correlation*

*<Image>*

*A screenshot of a cell phone

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*

Features that are highly correlated with other features are largely redundant and can be removed.

This task is iterated through a series of times and feeds into the final module that eliminates some of these highly correlated features.

*Exp2: Step 5. Remove Certain Highly Correlated Columns*

*A screenshot of a cell phone

Description automatically generated*

The Feature Engineering process brings the number of columns in the dataset up to 255, despite the removal of a sub-set of features.

Strings features have been converted into categorical values and further encoded into numbers so that the subsequent algorithms can build reliable models more efficiently.

### Experiment 3: Breakdown

*Breakdown of Experiment*

*Exp3: Step 1. Take Input from Experiment 2*

*A screenshot of a social media post

Description automatically generated*

Experiment 2 forms the basis for working out a feature selection list in Experiment 3.

*Exp3: Step 2. Split Data into Training and Test Data*

*A screenshot of a cell phone

Description automatically generated*

70% of the data is used to train the model. The remaining 30% is used to test the accuracy of the model. The Fraud data is split in proportion across the Train and Test dataset.

*Exp3: Step 3. Set up Algorithm for Modelling*

*A screenshot of a cell phone

Description automatically generated*

Two-Class Logistic Regression is the chosen algorithm. Parameters are left at the default assigned by Azure Machine Learning Studio.

*Exp3: Step 4. Set Up Training Module*

*A screenshot of a cell phone

Description automatically generated*

The label to predict against is the ‘Fraud’ column.

*Exp3: Step 5. Score Features by Importance*

*A screenshot of a cell phone

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*

The output of the Permutation Feature Importance module provides an ordered list of the features ranked by their importance in determining the scored result of the training credit card fraud predictive model.

This list will influence the column selection process as later experiments build up for the final production model.

### Experiment 4: Breakdown

*Breakdown of Experiment*

*Exp4: Step 1. Reload 25K Dataset – Apply Feature Selection*

*A screenshot of a social media post

Description automatically generated*

Out of the original 380 columns, we select 28 on which to train the predict credit card fraud model.

*Exp4: Step 2. LHS – Train Model with NO Feature Engineering*

*A close up of a map

Description automatically generated*

*Exp4: Step 3. RHS – Train Model with Feature Engineering*

*A picture containing screenshot

Description automatically generated*

The separate training modules are set up so that we can see the difference to performance metrics when Feature Engineering is applied and when it is not.

### Experiment 5: Breakdown

*Breakdown of Experiment*

*Exp5: Step 1. Feature Engineering (Done before ALL Modelling)*

*A screenshot of a cell phone

Description automatically generated*

The previous experiment has shown the benefit of Feature Engineering, which will now be applied to all experiments going forward.

*Exp5: Step 2. Train Model – Single Data Split / No Tuning*

*A close up of a map

Description automatically generated*

*Exp5: Step 3. Train Model –Cross Validation / Tuning*

*A close up of a map

Description automatically generated*

The purpose of this experiment is to show the benefit to improved training of the credit card fraud model by applying Cross Validation and Hyperparameter routines to the process.

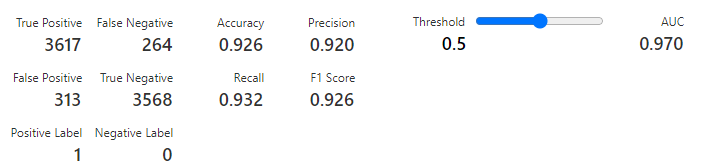
### Experiment 6: Breakdown

*Evaluation results for each classification algorithm*

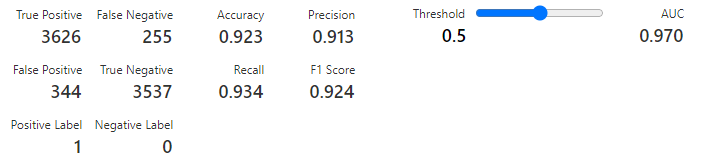
**Exp 6 – LHS**

**Two-Class Averaged Perceptron - v – Two-Class Boosted Decision Tree**

Two-Class Averaged Perceptron



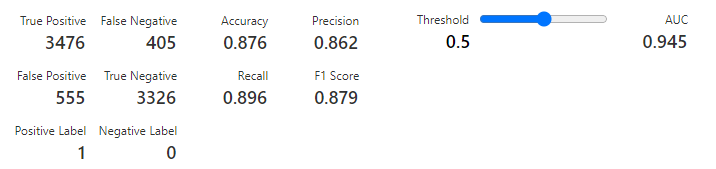
Two-Class Boosted Decision Tree



**Exp 6 – RHS**

**Two-Class Support Vector Machine - v – Two-Class Logistic Regression**

Two-Class Support Vector Machine



Two-Class Logistic Regression



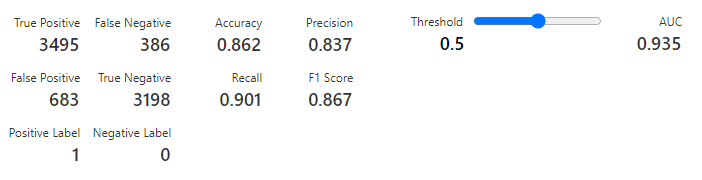
### Experiment 7: Breakdown

*Evaluation results for each classification algorithm*

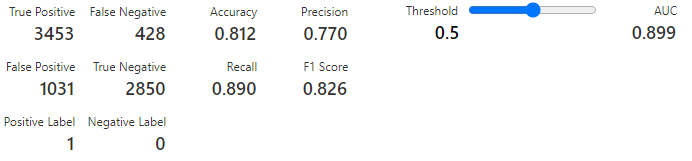
**Exp 7 – LHS**

**Two-Class Decision Forest - v – Two-Class Decision Jungle**

Two-Class Decision Forest



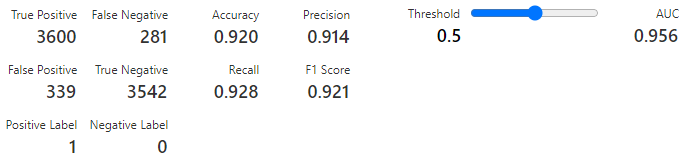
Two-Class Boosted Decision jungle



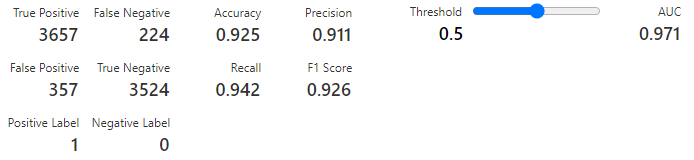
**Exp 7 – RHS**

**Two-Class Locally Deep Support Vector Machine - v – Two-Class Neural Network**

Two-Class Locally Deep Support Vector Machine



Two-Class Neural Network



### Experiment 8: Breakdown

*Breakdown of Experiment*

*Exp8: Step 2. Repeat Earlier Experiments with Larger Dataset*

*A screenshot of a cell phone

Description automatically generated*

Earlier experiments are repeated for Feature Engineering on the credit card fraud dataset, but this time on the full dataset of 25K records.

### Experiment 9: Breakdown

*Breakdown of Experiment*

*Exp9: Step 1. Extract 1% of Data – Train with 99%*

*A screenshot of a cell phone

Description automatically generated*

*A screenshot of a cell phone

Description automatically generated*

To train the model that will be deployed into production, 99% of the credit card fraud dataset is assigned to the training process.

The remaining 1% is taken to be used a ‘new’ data in the Shiny App UI. This data represents credit transactions that have not been used in the modelling process and can be considered ‘unseen’ data when the model is invoked through the project UI.

The ‘Split Data’ module is configured so that the proportion of Fraud/non-fraud records is maintained in both new datasets.

*Exp9: Step 2: Train Model based on Previous Experiments*

*A picture containing screenshot

Description automatically generated*

Applying all the learnings and outputs from previous experiments I have trained a credit card fraud predictive model ready for Production.

*Exp9: Step 3: Configure Web Service Input*

*A screenshot of a cell phone

Description automatically generated*

Once the Predictive experiment is created it is necessary to switch the Web service input so that it will pass in the requires 28 features, and not default to the original feature set of 380 from the starting dataset.

*Exp9: Step 4: Configure Web Service output*

A picture containing screenshot

Description automatically generated

The Predictive experiment condenses the Feature Engineering steps for numerical data into a single transformation and then applies the trained model.

The results of the model are then passed to the Web service output through which the data can be consumed by an external source.

## Credit Card Fraud Datasets

The datasets are too large to package in this document, but a sample CSV file is included with the final project submission.

# References / Bibliography – Final Report

Bhattacharyya, S., Jha, S., Tharakunnel, K. and Westland, J., 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), pp.602-613.

Ceronmani Sharmila, V., R., K., R., S., D., S. and R., H., 2019. Credit Card Fraud Detection Using Anomaly Techniques. *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, [online] 1(1), pp.1-4. Available at: <https://ieeexplore.ieee.org/document/8741421> [Accessed 11 September 2020].

Kurata, J., 2016. *Getting Started With Azure Machine Learning*. [online] App.pluralsight.com. Available at: <https://app.pluralsight.com/library/courses/azure-machine-learning-getting-started/table-of-contents> [Accessed 9 June 2020].

Lateef, Z., 2020. *Data Science Vs Machine Learning: What's The Difference? | Edureka*. [online] Edureka. Available at: <https://www.edureka.co/blog/data-science-vs-machine-learning/> [Accessed 9 September 2020].

Lima, R. and Pereira, A., 2017. Feature Selection Approaches to Fraud Detection in e-Payment Systems. *Lecture Notes in Business Information Processing*, [online] pp.111-126. Available at: <https://www.researchgate.net/publication/313731885\_Feature\_Selection\_Approaches\_to\_Fraud\_Detection\_in\_e-Payment\_Systems> [Accessed 11 September 2020].

Lucas, Y., Portier, P., Laporte, L., He, L., Caelen, O., Granitzer, M. and Calabretto, S., 2019. *Towards Automated Feature Engineering For Credit Card Fraud Detection Using Multi-Perspective Hmms*. 1st ed. [ebook] Lyon: Research Gate, pp.4-6. Available at: <https://www.researchgate.net/publication/335600419\_Towards\_automated\_feature\_engineering\_for\_credit\_card\_fraud\_detection\_using\_multi-perspective\_HMMs> [Accessed 1 September 2020].

Mahmoudi, N. and Duman, E., 2015. Detecting credit card fraud by Modified Fisher Discriminant Analysis. *Expert Systems with Applications*, [online] 42(5), pp.2510-2516. Available at: <https://www.semanticscholar.org/paper/Detecting-credit-card-fraud-by-Modified-Fisher-Mahmoudi-Duman/1cfb2a0a0f11dab8da5dc38d51a6e816f04ac8e3#paper-header> [Accessed 11 September 2020].

Microsoft, 2019. *ML Studio (Classic): Two-Class Averaged Perceptron - Azure*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-averaged-perceptron?redirectedfrom=MSDN> [Accessed 10 September 2020].

Microsoft, 2019. *ML Studio (Classic): Two-Class Support Vector Machine - Azure*. [online] Docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-support-vector-machine?redirectedfrom=MSDN> [Accessed 10 September 2020].

Narayanan, S., 2019. *Azure Machine Learning Studio Vs Azure Machine Learning Services | Codit*. [online] Codit. Available at: <https://www.codit.eu/blog/azure-machine-learning-studio-vs-services/?country\_sel=uk> [Accessed 9 September 2020].

Rhodes, J., 2020. *Building Your First Data Science Project In Microsoft Azure*. [online] App.pluralsight.com. Available at: <https://app.pluralsight.com/library/courses/microsoft-azure-building-first-data-science-project/table-of-contents> [Accessed 3 July 2020].

Sonobe, S., 2017. *Microstrategy Community*. [online] Community.microstrategy.com. Available at: <https://community.microstrategy.com/s/article/How-to-Visualize-Azure-Machine-Learning-Models-with-MicroStrategy?language=en\_US> [Accessed 10 September 2020].

Srinivasulu, R., 2019. *Preparing Data For Feature Engineering And Machine Learning In Microsoft Azure*. [online] App.pluralsight.com. Available at: <https://app.pluralsight.com/library/courses/microsoft-azure-preparing-data-feature-engineering-ml/table-of-contents> [Accessed 5 July 2020].