FRAUD DETECTION COMPLEX NEURAL NETWORK FOR FINANCIAL FRAUD DATA USING AWS SAGE MAKER CLOUD COMPUTING SOLUTIONS

1 EXECUTIVE SUMMARY

This report considers how Machine Learning can provide a solution for the issue of fraud such that affects COMPANY as a Share registrar and Pension service provider.

In the first part of the report AWS Sage Maker is outlined as a potential cloud architecture for creating a fraud detection model, with the second part of the report showcasing a fraud detection deep learning artificial neural network artifact created using public simulated dataset of credit card transactions, which establishes the principles and approach needed to apply similar model for COMPANY specific problems and datasets.

Data and privacy remain one of the key considerations and a plan for safe implementation within COMPANY is presented, whereby the PROD data could be used within COMPANY's own infrastructure and system to first generate an anonymized or even fully synthetic / simulated dataset. Once such a data model is acquired, it is uploaded to a Cloud Architecture of choice, where a model is trained, tested, validated, and adjusted until it produces satisfactory results. Such a model is then moved on-premises and can be used to tackle real-world scenarios using PROD data.

Implementation of such a fraud detection system would need to be divided into several stages, based on typical business use cases and complexity of the ML solution required.
First stage, which is showcased in the artifact using public dataset, would focus on transactions - as units for analysis - with the neural network looking at features of a transaction itself for patterns indicating fraud. Future stages / updates could see an Ensemble model (a model made up of many specialist models) take a holistic view of an account or entire dataset to spot Anomalies and/or patterns in the data and could be used to spot fraud type or fraudster gangs.

The artifact is developed in Google Collab using TensorFlow, with a baseline model contrasted against progressively mode advanced experiments, allowing for objective comparison of model's improving ability to predict and generalize. Focus is placed on data pre-processing, which is shown to have strong influence on the model's ability to learn patterns. The results achieved indicate that COMPANY will best be served by generating its own dataset and supplementing it with simulated data, where each feature can be thought through and controlled, so that the model has best chance to find patterns.

The volume of data is not at issue in such a scenario as Transfer Learning can be applied, where a small dataset can be used to fine-tune an established State of the Art model in the Fraud detection category to specialize it for COMPANY data.

2 TABLE OF CONTENTS

1 EXECUTIVE SUMMARY 1
2 Table of Contents 2
3 LIST OF TABLES 3
4 LIST OF FIGURES 3
5 GLOSSARY 3
6 Evaluation of the use of cloud computing for machine learning 4
6.1 Application Proposal 4
6.2 Platform Investigation and Critical Evaluation 4
7 Production of an artifact that demonstrates deep learning 7
7.1 Preamble 7
7.2 Data Description 7
7.2.1 Dataset 1: Credit Card Transactions Fraud Detection Dataset 7
7.2.2 Dataset 2: Synthetic Financial Dataset for Fraud Detection 8
7.3 Machine Learning Pipeline 8
7.4 Experimentation for Learning Optimisation 12
7.4.1 Base Model under sampled 12
7.4.2 Base Model Full dataset 13
7.4.3 Add a Hidden layer 14
7.4.4 Additional Epochs 16
7.5 Experimentation for Performance 17
7.6 Business Factors 18
7.7 Deployment Approach ..... **Error! Bookmark not defined.**
8 Personal reflection on the learning experience 19
9 Bibliography 19
10 APPENDIX A APRENTICESHIP STANDARDS ..... **Error! Bookmark not defined.**
11 APPENDIX B DATA ..... **Error! Bookmark not defined.**
12 AND SO ON WITH FURTHER APPENDICES AS NECESSARY … ..... **Error! Bookmark not defined.**

3 LIST OF TABLES

No table of figures entries found.

4 LIST OF FIGURES

Figure 1 - COMPANY AI Model development process using AWS SageMaker (step by step) Source: (Kulpa, 2024) Link 5
Figure 2 - Example of AWS SageMaker Architecture utilising many AWS and SageMaker services. Source: (Nandwani, Oyibo, & Shelton, 2022) link 6
Figure 3 - Base Model's Loss and Accuracy 12
Figure 4 - Confusion Matrix for base models 13
Figure 5 - model_0 (base model) architecture and training / fitting ..... **Error! Bookmark not defined.**
Figure 6 - model_0_b - architecture and fitting / trainingshown 14
Figure 7 - Accuracy and Loss for a Complex neural Network with 1 Hidden layer of 24 neurons ("relu") 15
Figure 8 - Confusion Matrix when Hidden Layer of 24 Neurons added ("relu") 15
Figure 9 - model_0_c architecture and fitting / training process shown 16
Figure 10 - model_0 (where pre-processing was controlled by us) with Hidden layer of 24 neurons ("relu") and given 30 epochs to train 17
Figure 11 - comparison of a simple fraud detection neural network with a single hidden layer with efficientnetb0 model 18

5 GLOSSARY

COMPANY – name of the company that is a Share Registrar and Pension Provider.

6 EVALUATION OF THE USE OF CLOUD COMPUTING FOR MACHINE LEARNING

6.1 APPLICATION PROPOSAL

As COMPANY moves into the AI era, a plethora of applications present themselves all of which would benefit from an early adoption by the COMPANY of the Cloud technologies such as are offered by e.g. Amazon's AWS. This report will investigate the Deep Learning aspect of the AI/ML solutions and propose a Fraud Detection Algorithm and a specific data secure way to develop it using AWS SageMaker in line with the COMPANY AI Policy and AI handbook.

The clear direction set by COMPANY's CEO to become a global share registrar and the focus placed on fraud prevention makes a Fraud Detection Algorithm a must have for COMPANY. Deep Learning Artificial Neural Networks have been proven extremely accurate in finding patterns and establishing decision boundaries for such problems. The app would be able to predict or raise a 'Red Flag' against ongoing transactions indicating possibility of fraud, ideally allowing COMPANY to act before fraud is committed. App would be held on-premises and be connected to an appropriate endpoint from where it could monitor ongoing transactions within current COMPANY infrastructure.

Please note that AI can help fight fraud in multiple ways (Lopez-Rojas & Axelsson, Money Laundering Detection using Synthetic Data, 2012) and with a focus on a specific part of the financial industry (Lopez-Rojas, Axelsson, & Gorton, RETSIM: A shoe store agent-based simulation for fraud detection, 2013). While this approach focuses on transaction data, an alternate approach, for example, would be a voice transcription algorithm that listens in on the incoming calls to detect fraudulent phrases or voice patterns and escalate the conversation to an experienced staff member.

COMPANY seems to approach the topic of AI with extreme caution, which is due to its responsibility as a share registrar and pension provider for the safety and privacy of its client data, which is why this report proposes to leverage Cloud Computing for Machine Learning only in so much as its necessary and only using synthetic / anonymised data (Lopez-Rojas E. , 2024).

6.2 PLATFORM INVESTIGATION AND CRITICAL EVALUATION

AWS SageMaker could be leveraged for model building (training and evaluation) using an already pre-processed, anonymised and synthesized dataset, which would be prepared on-premises and sent to SageMaker environment over secure connections.

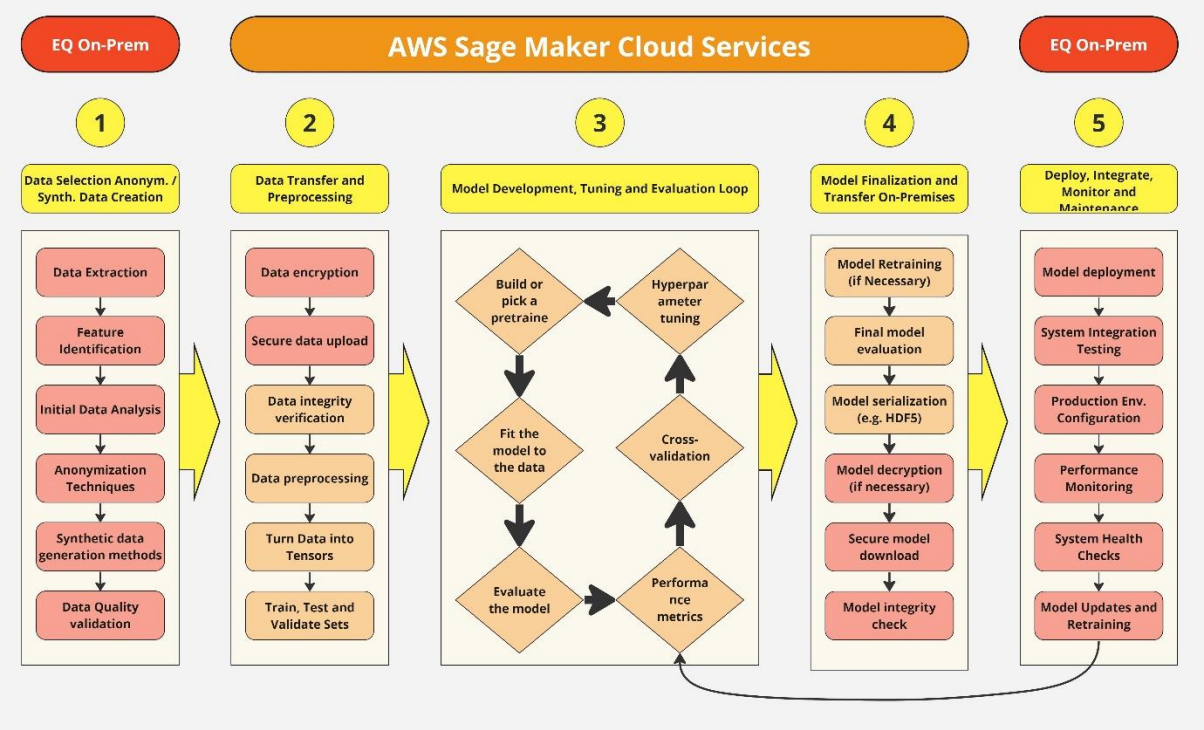

Figure 1 - COMPANY AI Model development process using AWS SageMaker (step by step) Source: (Kulpa, 2024) Link

Figure above shows a step-by-step process for how such an app could be developed using AWS SageMaker with a focus on data security as described previously.
AWS was chosen for this report, as it is already being implemented across COMPANY on some key PROJECTS, however same results can be achieved using competitors such as Google or Microsoft or even using on-premises hardware if significant ongoing investment was made in this area. Naturally, the difference is that AWS provides world-class ML capabilities on a per hour / on-demand basis, therefore significantly reducing costs of COMPANY's AI implementation (Amazon Inc, 2024).

As the previous figure shows AWS would be utilised especially for model building, training, and validation itself which are the aspects of Neural network design which require specialist hardware. If such a model was to be trained on a local device, such as a Dev laptop, model's training time could be days if not weeks (in a simple experiment using a relatively small dataset, the CPU training time per epoch was around 1:30 min, while a GPU training took 3 sec per epoch using a medium range, free access Google Collaboratory graphics card – Tesla T4). Since models inevitably grow in complexity as the business use case develops it would be best to setup a scalable infrastructure from on-set and prepare for the growth (Amazon Inc, 2024).

First COMPANY database structure for Share dealing and Pensions transactions must be analysed to establish the relevant features for the model to use, then PROD data extract with said features and a label identifying known fraud transactions could be performed. Data is then anonymized by removing any person specific details. Keep in mind that names, addresses and other confidential information are **NOT VIABLE** features for a model, so are not even part of the PROD dataset. Dataset could then be further synthesized by replicating it using a separate algorithm which replaces specifics of a given transaction with randomised / changed details while preserving the data structure and relationship between features. An example of such an approach is an external service provided by PaySim (Change, 2024), which learns the data model used by any payments related business to then generate fake transactions in keeping with the structure of the original dataset. Even if original transaction data were used, it would be turned into tensors which provides another degree of separation of the train/test dataset from the PROD data. An example of this is where the value of the transaction is turned into a

categorical variable of small, medium, large, exceptionally large, etc., depending on if the value e.g. falls between £0 - £1000.

It is unlikely that the fraud event is related to the specific amount being stolen, hence the actual amount might not be a valid feature. However, the comparative value of the transaction, e.g. as compared to average transactions for this account, could be used as part of feature engineering to bolster model's predictive capabilities.

Such a pre-processed dataset would then be moved over to AWS SageMaker for model generation. An artifact of such a model is created in part two of this report. Creation of the model would rely on AWS SageMaker's Jupyter Notebooks which allow for easy access to ML and visualization tools and the model can be coded in Python using TensorFlow or PyTorch with a variety of pre-built models available for Transfer Learning approach.
The fraud problem at hand requires a classification model, which can generate a non-linear decision boundary. The XGBoost model is a good example of AWS SageMaker's pre-built model offering that is specifically optimized for classification tasks and can handle imbalanced datasets which is a common feature of fraud detection scenarios (Amazon Inc, 2024).
As previously mentioned, the key feature of AWS SageMaker is its ability to quickly train the model using its infrastructure specifically designed for this task. Additionally, AWS offers Hyperparameter tuning, where these are automatically adjusted to optimize model's performance.

While this report suggests exporting the trained model from AWS to on-premises in line with data and ML policy at EQ, it is worth mentioning that AWS also offers Deployment and monitoring capabilities, which would allow for greater efficiency of the pipeline as the model develops and need retraining or updating.
It is worth mentioning that in an ideal scenario, should AWS be trusted with our data at some future point, development of such AI apps as this one can be done with relative ease even by those less experienced in AI/ML thanks to built-in Autopilot tools which guide the user through the development process.

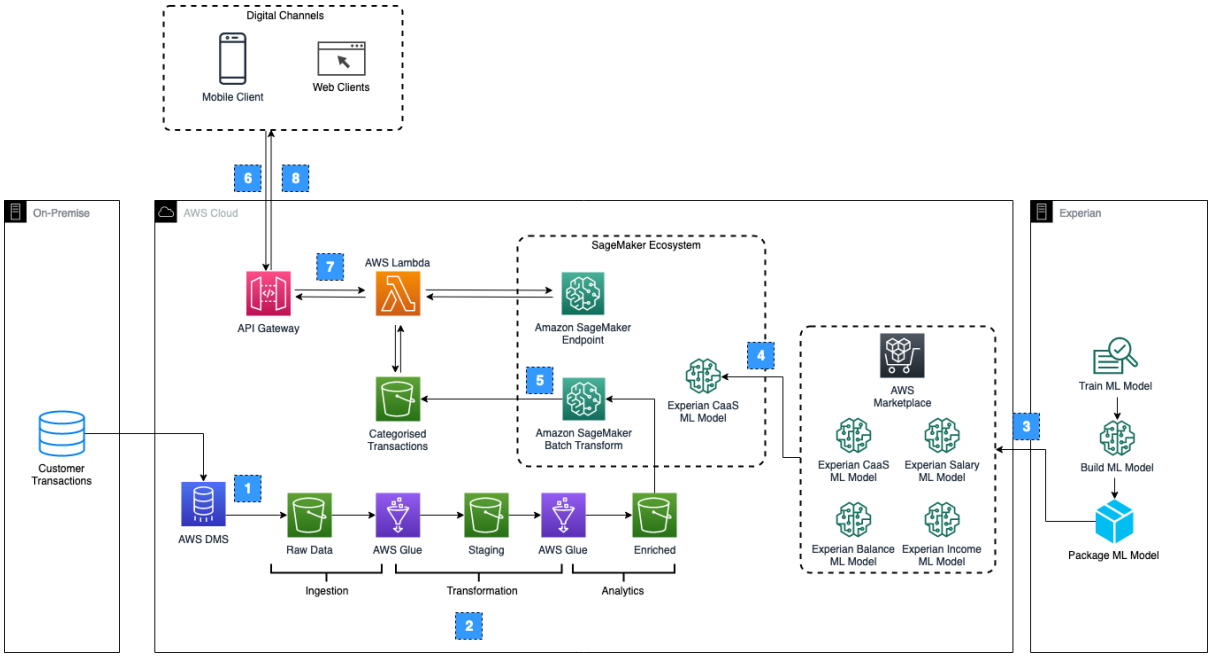

Figure 2 - Example of AWS SageMaker Architecture utilising many AWS and SageMaker services. Source: (Nandwani, Oyibo, & Shelton, 2022) link

7 PRODUCTION OF AN ARTIFACT THAT DEMONSTRATES DEEP LEARNING

7.1 PREAMBLE

This artifact will train and validate a neural network against two datasets that have previously been used to showcase Complex neural network's ability to find patterns in financial datasets. Artifact is developed in Google Collab using TensorFlow and in later stages shall be transferred over to AWS SageMaker should it be able to help tackle the problem of fraud.

7.2 DATA DESCRIPTION

The first step in developing an ML model is to find the correct data. COMPANY data is **not** used in this report, but two public datasets have been used that contain financial transactions and identified (labelled) fraud occurrences within.

7.2.1 Dataset 1: Credit Card Transactions Fraud Detection Dataset

This is a dataset generated by the Sparkov Data Generation tool (Shenoy, 2024). This dataset required pre-processing as it contains raw (simulated) customer data, which serves to exemplify how COMPANY will have to approach its data pre-processing. The simulator allows user to select 'profile' for data generation and this dataset contains a complex mix of transactions across all profiles to generate a more realistic representation.

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	state	zip	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat	merch_lon	is_fraud
0	01/01/2019 00:00	27031861	fraud_Rip	misc_net	4.97	Jennifer	Banks	F	561 Perry	Moravian F	NC	28654	36.0788	-81.178	3495	Psycholog	########	0b242abb	1.3E+09	36.0113	-82.048	0
1	01/01/2019 00:00	6.3E+11	fraud_Hell	grocery_p	107.23	Stephanie	Gill	F	43039 Rile	Orient	WA	99160	48.8878	-118.21	149	Special ed	########	1f76529f8	1.3E+09	49.15904	-118.19	0
2	01/01/2019 00:00	3.9E+13	fraud_Linc	entertainm	220.11	Edward	Sanchez	M	594 White	Malad City	ID	83252	42.1808	-112.26	4154	Nature cor	########	a1a22d70	1.3E+09	43.1507	-112.15	0
3	01/01/2019 00:01	35340937	fraud_Kut	gas_transp	45	Jeremy	White	M	9443 Cynt	Boulder	MT	59632	46.2306	-112.11	1939	Patent attc	########	6b849c16	1.3E+09	47.0343	-112.56	0
4	01/01/2019 00:03	3.8E+14	fraud_Keel	misc_pos	41.96	Tyler	Garcia	M	408 Bradle	Doe Hill	VA	24433	38.4207	-79.463	99	Dance mo	########	a41d7549	1.3E+09	38.675	-78.632	0
5	01/01/2019 00:04	47672653	fraud_Stro	gas_transp	94.63	Jennifer	Conner	F	4655 Davi	Dublin	PA	18917	40.375	-75.205	2158	Transport	########	189a841a	1.3E+09	40.6534	-76.153	0
6	01/01/2019 00:04	3E+13	fraud_Row	grocery_ne	44.54	Kelsey	Richards	F	889 Sarah	Holcomb	KS	67851	37.9931	-100.99	2691	Arboriculti	########	83ec1cc8	1.3E+09	37.16270	-100.15	0
7	01/01/2019 00:05	60113607	fraud_Con	gas_transp	71.65	Steven	Williams	M	231 Flores	Edinburg	VA	22824	38.8432	-78.6	6018	Designer,	########	6d294ed2	1.3E+09	38.9481	-78.54	0
8	01/01/2019 00:05	49227108	fraud_Her	misc_pos	4.27	Heather	Chase	F	6888 Hick	Manor	PA	15665	40.3359	-79.661	1472	Public affa	########	fc28024c	1.3E+09	40.3518	-79.958	0
9	01/01/2019 00:06	27208303	fraud_Sch	grocery_p	198.39	Melissa	Aguilar	F	21326 Tay	Clarksville	TN	37040	36.522	-87.349	151785	Patholog	########	3b9014ea	1.3E+09	37.1792	-87.485	0
10	01/01/2019 00:06	4.6E+12	fraud_Rutl	grocery_p	24.74	Eddie	Mendez	M	1831 Faitl	Clarinda	IA	51632	40.7491	-95.038	7297	IT trainer	########	d71c95ab	1.3E+09	40.27589	-96.012	0
11	01/01/2019 00:06	3.8E+14	fraud_Kerl	shopping_	7.77	Theresa	Blackwell	F	43576 Kris	Shenando	WV	25442	39.3716	-77.823	1925	Systems d	########	3c74776e	1.3E+09	40.1039	-78.624	0
12	01/01/2019 00:06	1.8E+14	fraud_Loc	grocery_p	71.22	Charles	Robles	M	3337 Lisa	Saint Pete	FL	33710	27.7898	-82.724	341043	Engineer, I	########	c1d9a7dd	1.3E+09	27.6306	-82.309	0
13	01/01/2019 00:07	55598574	fraud_Kiel	grocery_p	96.29	Jack	Hill	M	5916 Susa	Grenada	CA	96038	41.6125	-122.53	589	Systems a	########	413636e7	1.3E+09	41.6575	-122.23	0
14	01/01/2019 00:09	35148655	fraud_Beie	shopping_	7.77	Christoph	Castaneda	M	1632 Cohi	High Rolls	NM	88325	32.9396	-105.82	899	Naval arch	########	8a6293af	1.3E+09	32.8633	-106.52	0
15	01/01/2019 00:09	60119996	fraud_Sch	shopping_	3.26	Ronald	Carson	M	870 Roche	Harrington	NJ	7640	40.9918	-73.98	4664	Radiograp	########	baae0b09	1.3E+09	41.8312	-74.336	0
16	01/01/2019 00:10	60118602	fraud_Leb	misc_net	327	Lisa	Mendez	F	44259 Bet	Lahoma	OK	73754	36.385	-98.073	1078	Programm	########	991c0480	1.3E+09	36.38409	-99.048	0
17	01/01/2019 00:10	35654233	fraud_May	shopping_	341.67	Nathan	Thomas	M	4923 Cam	Carlisle	IN	47838	38.9763	-87.367	4081	Energy eng	########	f12cf52be	1.3E+09	38.67449	-88.306	0
18	01/01/2019 00:11	23482450	fraud_Kon	food_dinir	63.07	Justin	Gay	M	268 Hayes	Harborcre	PA	16421	42.1767	-79.942	2518	Event orga	########	8500f3d4	1.3E+09	41.4303	-79.493	0
19	01/01/2019 00:12	49568289	fraud_Sch	grocery_p	44.71	Kenneth	Robinson	M	269 Sanch	Elizabeth	NJ	7208	40.6747	-74.224	124967	Operation	########	09eff9c80	1.3E+09	40.0796	-74.848	0
20	01/01/2019 00:13	44697771	fraud_Bau	grocery_p	57.34	Gregory	Graham	M	4005 Dani	Methuen	MA	1844	42.728	-71.181	47249	Market res	########	139a1bee	1.3E+09	42.2688	-71.217	0
21	01/01/2019 00:14	23053366	fraud_Har	gas_transp	50.79	Jeffrey	Rice	M	21447 Pon	Moulton	IA	52572	40.6866	-92.683	1132	Probation	########	b9c4615b	1.3E+09	40.1889	-91.955	0
22	01/01/2019 00:17	1.8E+14	fraud_Klin	grocery_ne	46.28	Mary	Wall	F	2481 Mills	Plainfield	NJ	7060	40.6152	-74.415	71485	Leisure ce	########	19e23c6a	1.3E+09	40.0219	-74.228	0
23	01/01/2019 00:17	6.3E+11	fraud_Pac	shopping_	9.55	Susan	Washingto	F	759 Erin M	May	TX	76857	31.9571	-98.966	1791	Corporate	########	c4b4daeb	1.3E+09	31.6264	-98.61	0
24	01/01/2019 00:18	44287809	fraud_Les	shopping_	22.95	Richard	Waters	M	7683 Nata	Waukesha	WI	53186	42.9993	-88.22	95015	Therapist,	########	d2432bfe	1.3E+09	43.37504	-89.055	0
25	01/01/2019 00:18	3.4E+14	fraud_Kun	misc_net	2.55	Jodi	Foster	F	551 Zacha	Bailey	NC	27807	35.8072	-78.089	6629	Call centre	########	abe0676c	1.3E+09	36.7499	-78.678	0
26	01/01/2019 00:20	3.7E+14	fraud_Dec	grocery_p	64.09	Daniel	Escobar	M	61390 Hai	Romulus	MI	48174	42.2203	-83.358	31515	Police offi	########	6f363661	1.3E+09	42.36042	-83.552	0
27	01/01/2019 00:21	43342305	fraud_Bru	misc_pos	6.85	Scott	Martin	M	7483 Navi	Freedom	WY	83120	43.0172	-111.03	471	Education	########	f3c43d33	1.3E+09	43.7537	-111.45	0
28	01/01/2019 00:22	42259901	fraud_Kun	grocery_p	90.22	Brian	Simpson	M	2711 Dura	Honokaa	HI	96727	20.0827	-155.49	4878	Physiothe	########	95826e3c	1.3E+09	19.56	-156.05	0

Figure 3 - Dataset 1 (requires pre-processing)

The synthetic approach to dataset creation introduces the possibility of bias in the data if the simulated data's setup be influenced by preconceptions about fraud. This suggests that only a representative or real sample of fraud data will ensure objectivity of our data.

The most important aspect of the data, however, is the features that are deemed to hold the pattern that the neural network is expected to find and learn. Identifiers within the dataset ("cc_num" and "trans_num") will need to be removed, and impact of geospatial data needs to be considered, e.g. latitude and longitude can be processed directly, but additional transformation might be required to represent cluster locations or meaningful groups of data.

While we can establish patterns for victims of fraud using customer's age, gender, or job, we need to consider that the fraudster account will contain made up information and might introduce bad data into our network. Ideally features might need to be stripped to only those that cannot be invalidated by the fraudster actions easily. It might be worth looking at transactions for a given account rather than in aggregate and in the context of said account consider the time between this and previous

transaction, distance of this transaction to the mean or median for the account. Without such consideration asking a neural network to learn a pattern based on someone's age, gender, city, job, or date of birth is like asking it to do palm reading and does not seem to provide a scientific basis for pattern discovery.

7.2.2 Dataset 2: Synthetic Financial Dataset for Fraud Detection

This is a dataset generated by the PaySim mobile money simulator (Lopez-Rojas E. , 2024). Data pre-processing has been managed by the author and the downloadable dataset is already in a normalized state. It contains thirty columns of values between 0 – 1.0 and the feature names are not identifiable, aside from the fraud label (Class). This dataset approximates the state of the COMPANY's PROD dataset when imported into AWS.

This dataset's fraud instances are where fraudulent agent aims to profit by taking control of accounts, then transferring all funds to another account and finally withdrawing cash. System also flagged transactions as fraud whenever big transfers between accounts were attempted (over 200 000 in a single transaction). Transactions identified as fraud were subsequently cancelled, therefore several features (OldbalanceOrg, newbalanceOrig, oldBalanceDest and newbalanceDest) had to be removed during pre-processing stage to prevent the model from learning how to discover fraud by looking at which transactions were cancelled and highlighting those.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698	0.363787	0.090794	-0.5516	-0.6178	-0.99139	-0.31117	1.468177	-0.4704	0.207971	0.025791	0.403993	0.251412	-0.01831	0.277838	-0.11047	0.066928	0.128539	-0.18911	0.133558	-0.02105	149.62	0
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102	-0.25543	-0.16697	1.612727	1.065235	0.489095	-0.14377	0.635558	0.463917	-0.1148	-0.18336	-0.14578	-0.06908	-0.22578	-0.63867	0.101288	-0.33985	0.16717	0.125895	-0.00898	0.014724	2.69	0
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676	-1.51465	0.207643	0.624501	0.066084	0.717293	-0.16595	2.345865	-2.89008	1.109969	-0.12136	-2.26186	0.52498	0.247998	0.771679	0.909412	-0.68928	-0.32764	-0.1391	-0.05535	-0.05975	378.66	0
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436	-1.38702	-0.05495	-0.22649	0.178228	0.507757	-0.28792	-0.63142	-1.05965	-0.68409	1.965775	-1.23262	-0.20894	-0.1083	0.005274	-0.19032	-1.17558	0.647376	-0.22193	0.062723	0.061458	123.5	0
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053	0.817739	0.753074	-0.82284	0.538196	1.345852	-1.11967	0.175121	-0.45145	-0.23703	-0.03819	0.803487	0.408542	-0.00943	0.798278	-0.13746	0.141267	-0.20601	0.502292	0.219422	0.215153	69.99	0
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314	-0.56867	-0.37141	1.341262	0.359894	-0.35809	-0.13713	0.517617	0.401726	-0.05813	0.068653	-0.03319	0.084968	-0.20825	-0.55982	-0.0264	-0.37143	-0.23279	0.105915	0.253844	0.08108	3.67	0
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213	0.46496	-0.09925	-1.41691	-0.15383	-0.75106	0.167372	0.050144	-0.44359	0.002821	-0.61199	-0.04558	-0.21963	-0.16772	-0.27071	-0.1541	-0.78006	0.750137	-0.25724	0.034507	0.005168	4.99	0
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786	0.615375	1.249376	-0.61947	0.291474	1.757964	-1.32387	0.686133	-0.07613	-1.22213	-0.35822	0.324505	-0.15674	1.943465	-1.01545	0.057504	-0.64971	-0.41527	-0.05163	-1.20692	-1.08534	40.8	0
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084	-0.39205	-0.41043	-0.70512	-0.11045	-0.28625	0.074355	-0.32878	-0.21008	-0.49977	0.118765	0.570328	0.052736	-0.07343	-0.26809	-0.20423	1.011592	0.373205	-0.38416	0.011747	0.142404	93.2	0
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539	-0.73673	-0.36685	1.017614	0.83639	1.006844	-0.44352	0.150219	0.739453	-0.54098	0.476677	0.451773	0.203711	-0.24691	-0.63375	-0.12079	-0.38505	-0.06973	0.094199	0.246219	0.083076	3.68	0
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456	-1.72041	1.626659	1.199644	-0.67144	-0.51395	-0.09505	0.23093	0.031967	0.253415	0.854344	-0.22137	-0.38723	-0.0093	0.313894	0.02774	0.500512	0.251367	-0.12948	0.04285	0.016253	7.8	0
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247	-0.55889	0.309755	-0.25912	-0.32614	-0.09005	0.362832	0.928904	-0.12949	-0.80998	0.359985	0.707664	0.125892	0.049924	0.238422	0.00913	0.99671	-0.76731	-0.49221	0.042472	-0.05434	9.99	0
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749	-2.09401	1.323729	0.227666	-0.24268	1.205417	-0.31763	0.725675	-0.81561	0.873936	-0.84779	-0.68319	-0.10276	-0.23181	-0.48329	0.084668	0.392831	0.161135	-0.35499	0.026416	0.042422	121.5	0
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982	-0.22108	0.46023	-0.77366	0.323387	-0.01108	-0.17849	-0.65556	-0.19993	0.124005	-0.9805	-0.98292	-0.1532	-0.03688	0.074412	-0.07141	0.104744	0.548265	0.104094	0.021491	0.021293	27.5	0
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711	0.755713	1.151087	0.844555	0.792944	0.370448	-0.73498	0.406796	-0.30306	-0.15587	0.778265	0.221868	-1.58212	1.151663	0.222182	1.020586	0.028317	-0.23275	-0.23556	-0.16478	-0.03015	58.8	0
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603	-0.43617	0.747731	-0.79398	-0.77041	1.047627	-1.0666	1.106953	1.660114	-0.27927	-0.41999	0.432535	0.263451	0.499625	1.35365	-0.25657	-0.06508	-0.03912	-0.08709	-0.181	0.129394	15.99	0
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938	0.782333	-0.26798	-0.45031	0.936708	0.70838	-0.46865	0.354574	-0.24663	-0.00921	-0.59591	-0.57568	-0.11391	-0.02461	0.196002	0.013802	0.103758	0.364298	-0.38226	0.092809	0.037051	12.99	0
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962	-0.66527	-0.73798	0.324098	0.277192	0.252624	-0.2919	-0.18452	1.143174	-0.92871	0.68047	0.025436	-0.04702	-0.1948	-0.67264	-0.15686	-0.88839	-0.34241	-0.04903	0.079692	0.131024	0.89	0
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842	1.23309	0.345173	0.91723	0.970117	-0.26657	-0.47913	-0.52661	0.472004	-0.72548	0.075081	-0.40687	-2.19685	-0.5036	0.98446	2.458589	0.042119	-0.48163	-0.62127	0.392053	0.949594	46.8	0
15	1.492936	-1.02935	0.454795	-1.43803	-1.55543	-0.72096	-1.08966	-0.05313	-1.97868	1.638076	1.077542	-0.63205	-0.41696	0.052011	-0.04298	-0.16643	0.304241	0.554432	0.85423	-0.38791	-0.17765	-0.17507	0.040002	0.295814	0.332911	-0.22038	0.022298	0.007602	5	0
16	0.694885	-1.36182	1.029221	0.834159	-1.19121	1.309109	-0.87859	0.44529	-0.4462	0.568521	1.019151	1.298329	0.42048	-0.37265	-0.80798	-2.04456	0.515663	0.625847	-1.30041	-0.13833	-0.29558	-0.57196	-0.05088	-0.30421	0.072001	-0.42223	0.086553	0.063499	231.71	0
17	0.962496	0.328461	-0.17148	2.109204	1.129566	1.696038	0.107712	0.521502	-1.19131	0.724396	1.69033	0.406774	-0.93642	0.983739	0.710911	-0.60223	0.402484	-1.73716	-2.02761	-0.26932	0.143997	0.402492	-0.04851	-1.37187	0.390814	0.199964	0.016371	-0.01461	34.09	0
18	1.166616	0.50212	-0.0673	2.261569	0.428804	0.089474	0.241147	0.138082	-0.98916	0.922175	0.744786	-0.53138	-2.10535	1.12687	0.003075	0.424425	-0.45448	-0.09887	-0.8166	-0.30717	0.018702	-0.06197	-0.10385	-0.37042	0.6032	0.108556	-0.04052	-0.01142	2.28	0
18	0.247491	0.277666	1.185471	-0.0926	-1.31439	-0.15012	-0.94636	-1.61794	1.544071	-0.82988	-0.5832	0.524933	-0.45338	0.081393	1.555204	-1.39689	0.783131	0.436621	2.177807	-0.23098	1.65018	0.200454	-0.18535	0.423073	0.820591	-0.22763	0.336634	0.250475	22.75	0
22	-1.94653	-0.0449	-0.40557	-1.01306	2.941968	2.955053	-0.06306	0.855546	0.049967	0.573743	-0.08126	-0.21575	0.044161	0.033898	1.190718	0.578843	-0.97567	0.044063	0.488603	-0.21672	-0.57963	-0.79923	0.8703	0.983421	0.321201	0.14965	0.707519	0.0146	0.89	0
22	-2.07429	-0.12148	1.322021	0.410008	0.295198	-0.95954	0.543985	-0.10463	0.475664	0.149451	-0.85657	-0.18052	-0.65523	-0.2798	-0.21167	-0.33332	0.010751	-0.48847	0.505751	-0.38669	-0.40364	-0.2274	0.742435	0.398535	0.249212	0.274404	0.359969	0.243232	26.43	0
23	1.173285	0.353498	0.283985	1.133563	-0.17258	-0.91605	0.369025	-0.32726	-0.24665	-0.04614	-0.14342	0.97935	1.492285	0.101418	0.761478	-0.01458	-0.51164	-0.32506	-0.38093	0.027878	0.067003	0.227812	-0.15049	0.435945	0.724825	-0.33708	0.016368	0.039041	41.88	0
23	1.322707	-0.17404	0.434555	0.576038	-0.83676	-0.83108	-0.2649	-0.22098	-1.07142	0.868559	-0.64151	-0.11132	0.361485	0.171945	0.782167	-1.35587	-0.21694	1.271765	-1.24062	-0.52295	-0.28438	-0.32336	-0.03771	0.347151	0.559639	-0.28016	0.042335	0.028822	16	0
23	-0.41429	0.905437	1.727453	1.473471	0.007443	-0.20033	0.740228	-0.02925	-0.59339	-0.34619	-0.01214	0.786796	0.635954	-0.08632	0.076804	-1.40592	0.775592	-0.94289	0.543969	0.097308	0.077237	0.457331	-0.0385	0.642522	-0.18389	-0.27746	0.182687	0.152665	33	0
23	1.059387	-0.17532	1.26613	1.18611	-0.786	0.578435	-0.76708	0.401046	0.6995	-0.06474	1.048292	1.005618	-0.542	-0.03991	-0.21868	0.004476	-0.19355	0.042388	-0.27783	-0.17802	0.013676	0.213734	0.014462	0.002951	0.294638	-0.39507	0.081461	0.02422	12.99	0
24	1.237429	0.061043	0.380526	0.761564	-0.35977	-0.49408	0.006494	-0.13386	0.43881	-0.20736	-0.92918	0.527106	0.348676	-0.15254	-0.21839	-0.19155	-0.11658	-0.63379	0.348416	-0.06635	-0.24568	-0.5309	-0.04427	0.079168	0.509136	0.288858	-0.0227	0.011836	17.28	0
25	1.114009	0.085546	0.493702	1.33576	-0.30019	-0.01075	-0.11876	0.186617	0.205687	0.082262	1.133556	0.626699	-1.49278	0.520788	-0.67459	-0.52911	0.158256	-0.39875	-0.14571	-0.27383	-0.05323	-0.00476	-0.03147	0.198054	0.565007	-0.33772	0.029057	0.004453	4.45	0
26	-0.52991	0.873892	1.347247	0.145457	0.414209	0.100223	0.711206	0.176066	-0.28672	-0.48469	0.87249	0.851636	-0.57175	0.100974	-1.51977	-0.28438	-0.31052	-0.40425	-0.82337	-0.29035	0.046949	0.208105	-0.18555	0.001031	0.098816	-0.5529	-0.07329	0.023307	6.14	0
26	-0.52991	0.873892	1.347247	0.145457	0.414209	0.100223	0.711206	0.176066	-0.28672	-0.48469	0.87249	0.851636	-0.57175	0.100974	-1.51977	-0.28438	-0.31052	-0.40425	-0.82337	-0.29035	0.046949	0.208105	-0.18555	0.001031	0.098816	-0.5529	-0.07329	0.023307	6.14	0
26	-0.53539	0.865268	1.351076	0.147575	0.43368	0.086983	0.693039	0.179742	-0.28564	-0.48247	0.8718	0.853447	-0.57182	0.102252	-1.51999	-0.28591	-0.30963	-0.4039	-0.82374	-0.28326	0.049526	0.206537	-0.18711	0.000753	0.098117	-0.55347	-0.07831	0.025427	1.77	0
26	-0.53539	0.865268	1.351076	0.147575	0.43368	0.086983	0.693039	0.179742	-0.28564	-0.48247	0.8718	0.853447	-0.57182	0.102252	-1.51999	-0.28591	-0.30963	-0.4039	-0.82374	-0.28326	0.049526	0.206537	-0.18711	0.000753	0.098117	-0.55347	-0.07831	0.025427	1.77	0
27	-0.24605	0.473267	1.695738	0.262411	-0.01087	-0.61084	0.793937	-0.24725	0.138879	-0.40101	-0.81205	-0.18352	-0.63011	-0.28622	-0.33705	-0.42811	-0.0306	-0.50457	0.35518	0.045721	-0.1946	-0.33513	-0.07818	0.392784	-0.03116	0.198037	-0.17535	-0.20362	30.49	0
27	-1.45219	1.765124	0.611669	1.176825	-0.44598	0.246826	-0.25757	1.092472	-0.60752	0.047156	0.783727	1.096386	-0.26809	0.768648	-0.52437	-0.80882	0.710386	-0.11837	0.979288	0.008713	0.08228	0.325782	-0.06911	0.020962	-0.04467	-0.24344	0.14918	0.120557	1.8	0
29	0.99637	-0.12259	0.546819	0.70658	0.13456	1.156995	-0.29456	0.407429	0.337863	-0.40815	0.682646	1.25372	0.391917	-0.03751	0.796302	-1.66821	1.322995	-2.83004	-1.46343	-0.20346	-0.0763	0.108867	0.162231	-0.57562	0.109795	0.373813	0.050552	0.005106	20.53	0
29	1.11088	0.168717	0.517144	1.325407	-0.19157	0.019504	-0.03185	0.11762	0.017665	0.044865	1.345075	1.28634	-0.25227	0.274458	-0.81039	-0.58701	0.087451	-0.55047	-0.15475	-0.19012	-0.03771	0.095701	-0.0482	0.232115	0.606201	-0.3421	0.03677	0.00748	6.54	0

Figure 4 - Dataset 2 (already pre-processed by original author / third-party)

7.3 Machine Learning Pipeline

To preprocess dataset 1, we first drop columns which cannot serve as features in fraud detection ("cc_num", "trans_num"), those where word embedding is usually required (e.g. "job"), but is not part of our model design, or which introduce high dimensionality ("street", "city", "zip").

```python
from sklearn.preprocessing import StandardScaler, OneHotEncoder, FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.base import BaseEstimator, TransformerMixin
import datetime
import numpy as np
import pandas as pd

# Custom transformer to calculate age from DOB
class AgeTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X.applymap(lambda dob: datetime.datetime.now().year - datetime.datetime.strptime(dob, '%Y-%m-%d').year).values.reshape(-1, 1)

# Custom transformer to extract date time features
class DateTimeFeaturesTransformer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        dt_series = pd.to_datetime(X.iloc[:, 0], format='%Y-%m-%d %H:%M:%S')
        return np.vstack([dt_series.dt.hour, dt_series.dt.dayofweek]).T

train_data = train_data.drop(["cc_num", "trans_num", "street", "city", "zip", "job", "Unnamed: 0", "merchant", "first", "last"], axis=1)
test_data = test_data.drop(["cc_num", "trans_num", "street", "city", "zip", "job", "Unnamed: 0", "merchant", "first", "last"], axis=1)

# Separate the target variable before applying transformations
y_train = train_data["is_fraud"].values # Convert to numpy array
y_test = test_data["is_fraud"].values # Convert to numpy array

X_train = train_data.drop(["is_fraud"], axis=1)
X_test = test_data.drop(["is_fraud"], axis=1)
```

Figure 5 - Dataset 1 pre-processing code.

Then the continuous and categorical features are split, the continuous values are scaled (normalized) into range of 0-1, while the categorical features are one-hot encoded. Age is derived from Date of Birth and DateTime is transformed to make e.g. ‘day of the week’ or ‘hour’ a feature to be looked at by the neural network.

```python
from sklearn.preprocessing import OneHotEncoder

# Select the continuous columns to be scaled
continuous_cols = ['amt', 'lat', 'long', 'city_pop', 'merch_lat', 'merch_long']
# Select columns for one-hot encoding
categorical_cols = ['category', 'gender', 'state']  # Add other categorical columns as necessary

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, continuous_cols),
        ('cat', categorical_transformer, categorical_cols),
        # For high cardinality features, you could use a different strategy, e.g., feature hashing
        ('age', AgeTransformer(), ['dob']),
        ('dt', DateTimeFeaturesTransformer(), ['trans_date_trans_time'])
    ],
    remainder='drop')

# Fit on training data and transform both training and test data
train_features = preprocessor.fit_transform(X_train)
test_features = preprocessor.transform(X_test)
```

Figure 6 - Dataset 1 pre-processing code continued

These are then turned into a tensor and saved as X_train, y_train, X_test and y_test (Dataset 1 came already split into train and test with a 75/25 ratio).

Data then needs to be resampled due to the heavily unbalanced nature of fraud datasets. The RandomUnderSampler from the Imblearn.under_sampling library is used to achieve a 2:1 ratio of non-fraud to fraud transactions. It takes snippets from the non-fraud transactions until a preset ratio is achieved, as shown below.

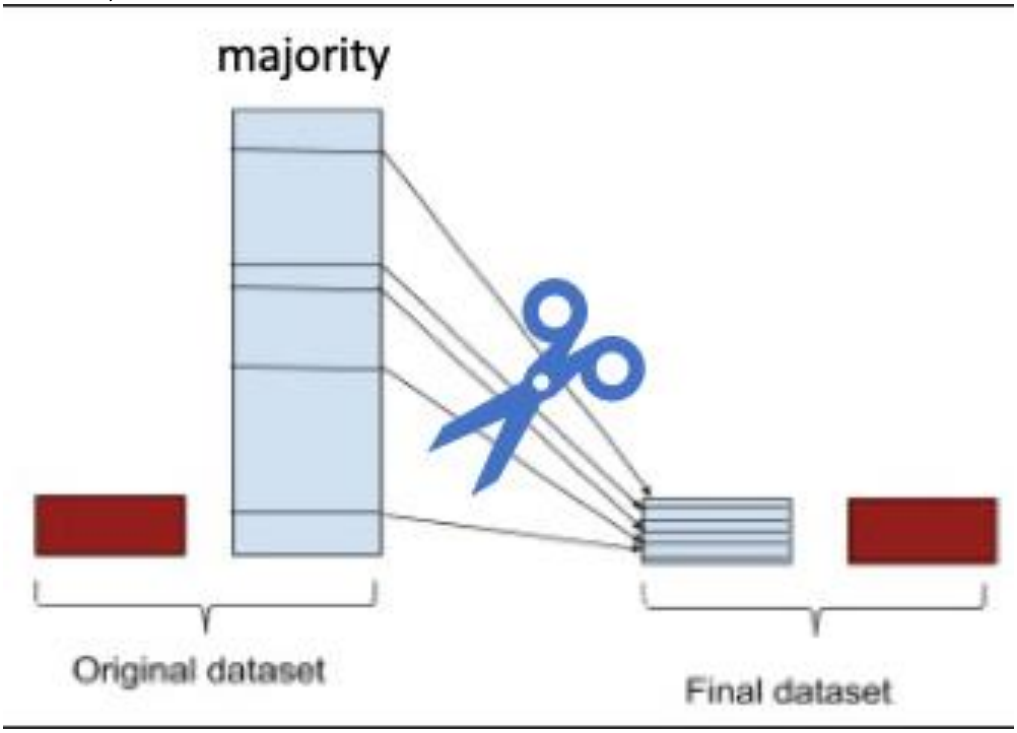

Figure 7 - How Under Sampling works.

```
[6]  # Random Undersampling

     import imblearn
     from imblearn.under_sampling import RandomUnderSampler

     undersample = RandomUnderSampler(sampling_strategy=0.5)

     X_train_under, y_train_under = undersample.fit_resample(X_train, y_train)

     X_test_under, y_test_under = undersample.fit_resample(X_test, y_test)
```

Figure 8 - Under sampling Code.

Visualizations of data are mostly handled by matplotlib.pyplot and seaborn libraries, with some functions saved within an external helper_functions.py file for reproducibility and clarity.

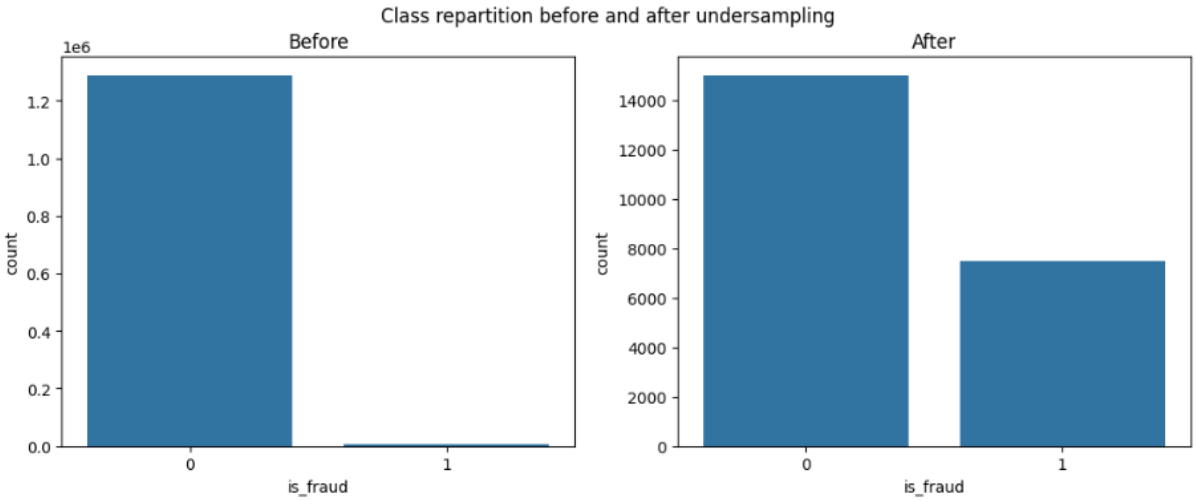

Figure 9 - Visualization of dataset before and after Under sampling.

A base-model is created to benchmark future optimisations against, using TensorFlow Sequential API with a basic twenty-four neuron input layer utilising “relu” activation and a single neuron output layer with “sigmoid” activation. Binary cross entropy has been used for this classification task and a jack-of-all-trades optimizer Adam was applied. A tensorboard callback will generate data for further analysis, however two visualizations are generated for each model featuring loss and accuracy curves and a confusion matrix.

```
[16]
import tensorflow as tf
import helper_functions

model_0 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(24, activation="relu"),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model_0.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                optimizer=tf.keras.optimizers.Adam(),
                metrics=["accuracy"])
```

```
[17] history_0 = model_0.fit(X_train_under, y_train_under,
                             callbacks=[helper_functions.create_tensorboard_callback("tensorboard","model_0")],
                             epochs=10,
                             validation_data=(X_test_under, y_test_under),
)
```

```
Saving TensorBoard log files to: tensorboard/model_0/20240325-092109
Epoch 1/10
704/704 [==============================] - 4s 4ms/step - loss: 0.4019 - accuracy: 0.8632 - val_loss: 0.3568 - val_accuracy: 0.8802
Epoch 2/10
704/704 [==============================] - 2s 2ms/step - loss: 0.3211 - accuracy: 0.8861 - val_loss: 0.3424 - val_accuracy: 0.8657
Epoch 3/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2989 - accuracy: 0.8874 - val_loss: 0.3080 - val_accuracy: 0.8799
Epoch 4/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2863 - accuracy: 0.8894 - val_loss: 0.3030 - val_accuracy: 0.8878
Epoch 5/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2796 - accuracy: 0.8894 - val_loss: 0.2972 - val_accuracy: 0.8758
Epoch 6/10
704/704 [==============================] - 2s 3ms/step - loss: 0.2734 - accuracy: 0.8896 - val_loss: 0.2918 - val_accuracy: 0.8777
Epoch 7/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2671 - accuracy: 0.8884 - val_loss: 0.2813 - val_accuracy: 0.8819
Epoch 8/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2603 - accuracy: 0.8915 - val_loss: 0.2930 - val_accuracy: 0.8923
Epoch 9/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2566 - accuracy: 0.8905 - val_loss: 0.2717 - val_accuracy: 0.8811
Epoch 10/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2518 - accuracy: 0.8913 - val_loss: 0.2768 - val_accuracy: 0.8844
```

Figure 10 - base model architecture and training / fitting process.

7.4 EXPERIMENTATION FOR LEARNING OPTIMISATION

7.4.1 Base Model under sampled

The optimization was divided into two pillars. model_0 is the base model utilizing Dataset 1 which underwent pre-processing, which itself could have introduced errors. Therefore model_1 was created which had the same structure as model_0 but was fitted against dataset 2, which was pre-processed by the dataset's author already (and so assumed to work for ML).

Furthermore, each base model features sub-models (a, b, c) each attempting to improve the scores by introducing a single modification.

First observation for both base models is that the validation accuracy curve, although moving in the right direction, suffers from volatile changes. This could indicate that the neural network is struggling to learn the pattern within the data, that the pattern is conflicting or that the data requires reshuffling.

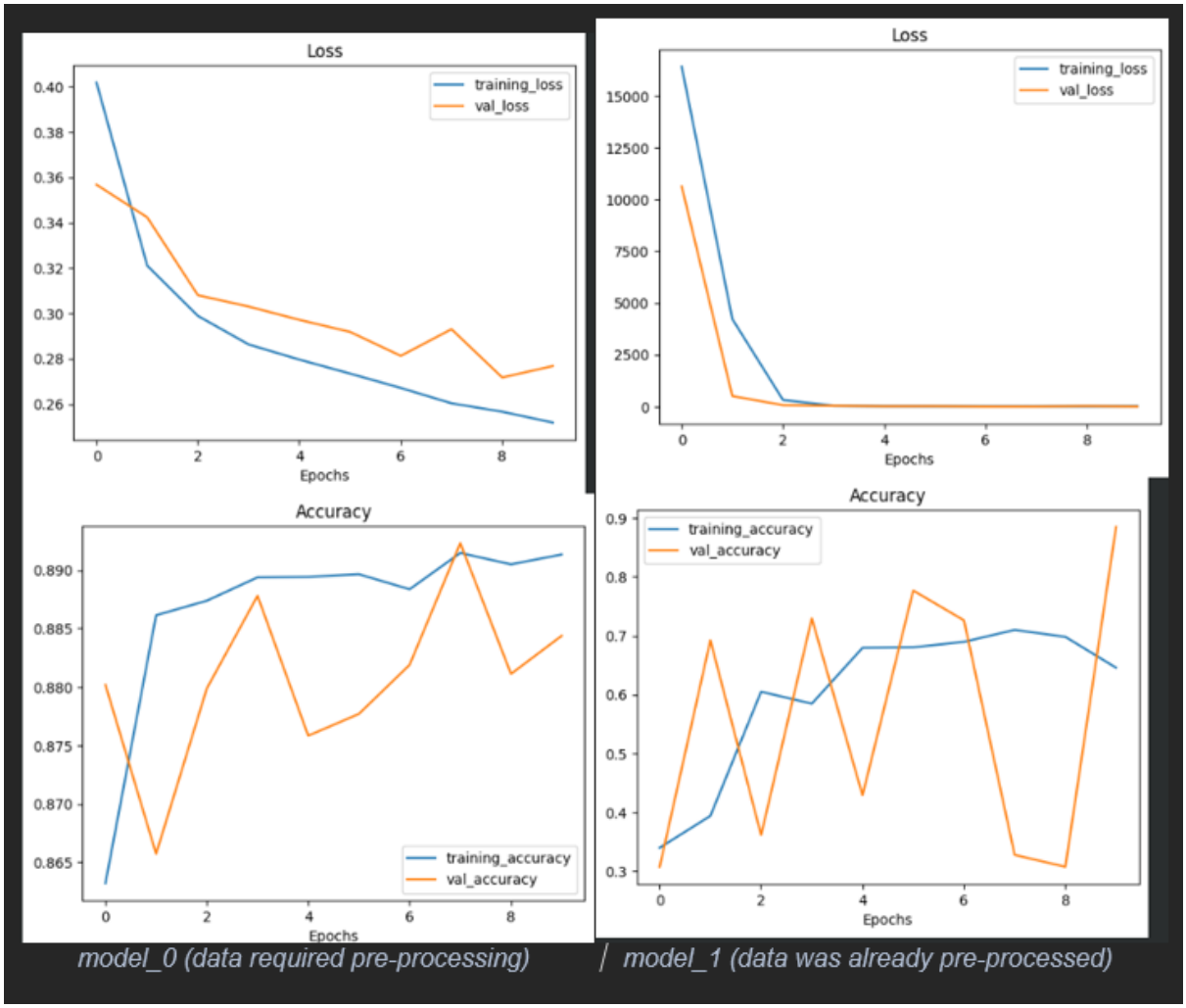

Figure 11 - Base Model's Loss and Accuracy

Layers: input twenty-four neurons RELU, no hidden layer, output 1 neuron Sigmoid

loss = BinaryCrossentropy, Optimizer = Adam

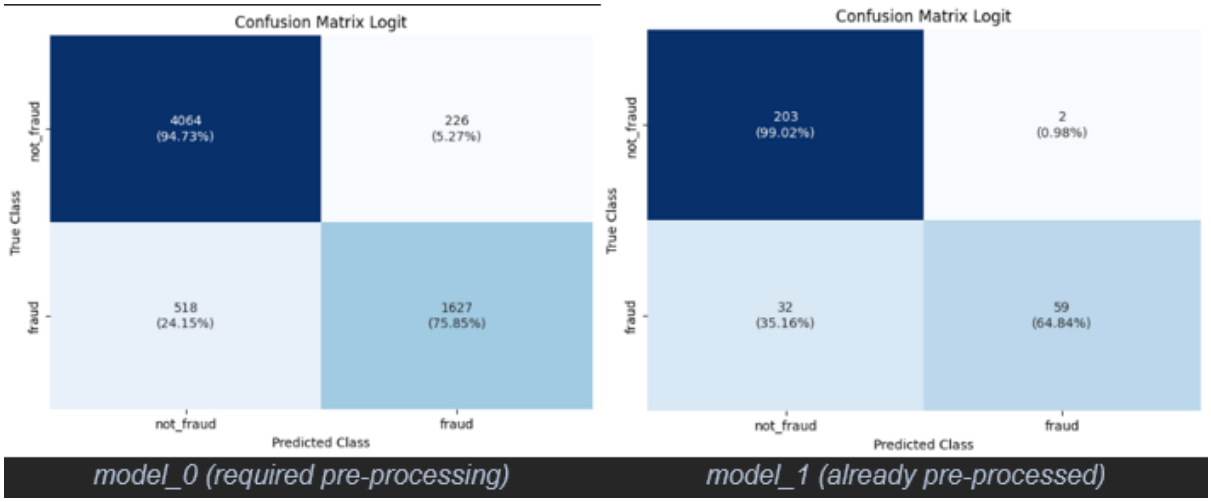

Figure 12 - Confusion Matrix for base models.

Layers: input twenty-four neurons RELU, no hidden layer, output 1 neuron Sigmoid

loss = BinaryCrossentropy, Optimizer = Adam

7.4.2 Base Model Full dataset

While it is recommended to under sample fraud datasets to make the classes more balanced, it only makes sense to expect the model to then be able to generalize across the entire dataset. The below figures show a confusion matrix for the full datasets (generating loss curves would involve fitting the model against the full dataset which would obscure the patterns learned, if any).

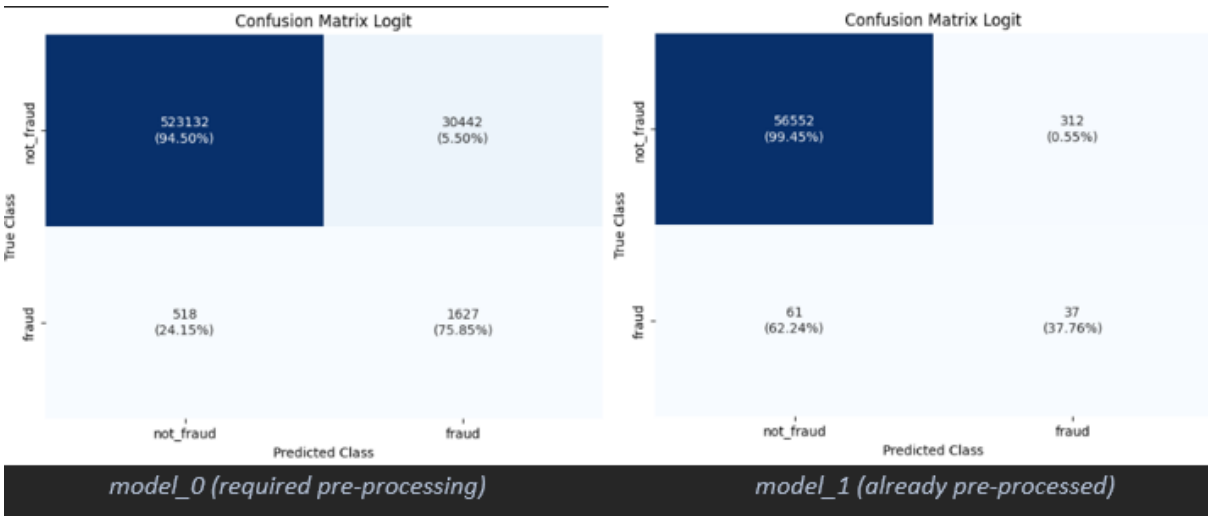

Figure 5 - Confusion Matrix for Base Models using FULL dataset (not under sampled)

The confusion matrix shows that the model_0 for which we controlled the pre-processing process does generalize for the entire dataset, while model_1 where pre-processing has been performed by a third party (author of the simulated dataset) looses on accuracy considerably, generating multiple false negatives (fraud instances that were predicted as not being fraud).

7.4.3 Adding a Hidden layer

Addition of a hidden layer containing twenty-four neurons (“relu” activation) did improve the accuracy of the model_0, but has cause significant overfitting for model_1 which was not able to predict when working with even the under sampled test dataset:

```
model_0_b = tf.keras.models.Sequential([
    tf.keras.layers.Dense(24, activation="relu"),
    tf.keras.layers.Dense(24, activation="relu"),
    tf.keras.layers.Dense(1, activation="sigmoid")
])

model_0_b.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                  optimizer=tf.keras.optimizers.Adam(),
                  metrics=["accuracy"])
```

```
[47] history_0_b = model_0_b.fit(X_train_under, y_train_under,
                                 callbacks=[helper_functions.create_tensorboard_callback("tensorboard","model_0_b")],
                                 epochs=10,
                                 validation_data=(X_test_under, y_test_under),
)
```

```
Saving TensorBoard log files to: tensorboard/model_0_b/20240325-111708
Epoch 1/10
704/704 [==============================] - 6s 3ms/step - loss: 0.4336 - accuracy: 0.8351 - val_loss: 0.3428 - val_accuracy: 0.8768
Epoch 2/10
704/704 [==============================] - 2s 3ms/step - loss: 0.3059 - accuracy: 0.8886 - val_loss: 0.2970 - val_accuracy: 0.8833
Epoch 3/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2762 - accuracy: 0.8914 - val_loss: 0.2904 - val_accuracy: 0.8822
Epoch 4/10
704/704 [==============================] - 2s 2ms/step - loss: 0.2630 - accuracy: 0.8920 - val_loss: 0.2875 - val_accuracy: 0.8817
Epoch 5/10
704/704 [==============================] - 3s 4ms/step - loss: 0.2500 - accuracy: 0.8934 - val_loss: 0.2644 - val_accuracy: 0.8842
Epoch 6/10
704/704 [==============================] - 2s 3ms/step - loss: 0.2268 - accuracy: 0.9037 - val_loss: 0.2288 - val_accuracy: 0.9047
Epoch 7/10
704/704 [==============================] - 2s 3ms/step - loss: 0.2019 - accuracy: 0.9160 - val_loss: 0.2124 - val_accuracy: 0.9110
Epoch 8/10
704/704 [==============================] - 2s 2ms/step - loss: 0.1814 - accuracy: 0.9268 - val_loss: 0.1936 - val_accuracy: 0.9203
Epoch 9/10
704/704 [==============================] - 2s 3ms/step - loss: 0.1650 - accuracy: 0.9341 - val_loss: 0.1781 - val_accuracy: 0.9279
Epoch 10/10
704/704 [==============================] - 2s 3ms/step - loss: 0.1524 - accuracy: 0.9409 - val_loss: 0.1743 - val_accuracy: 0.9301
```

Figure 13 - model_0_b - architecture and fitting / training shown.

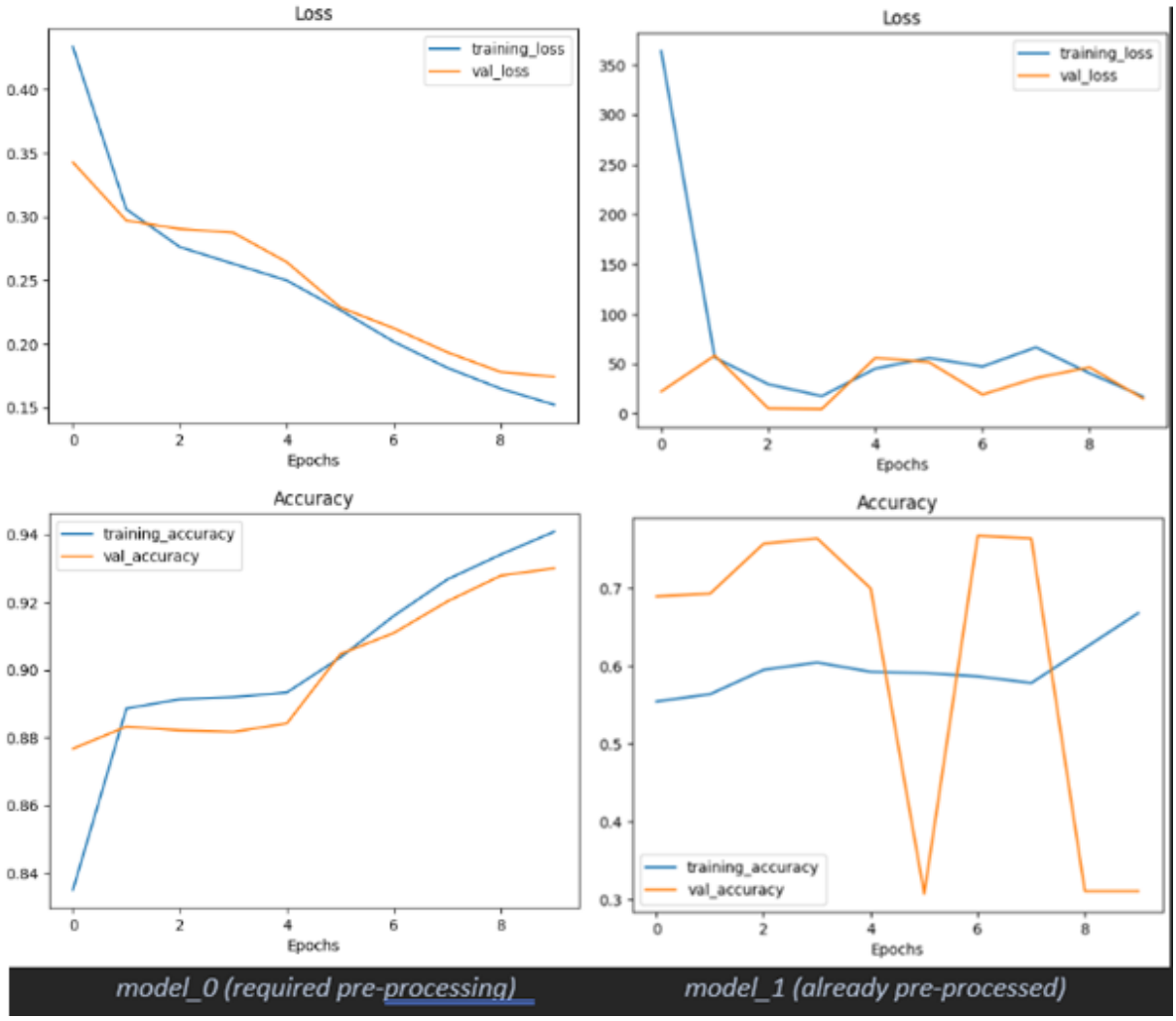

Figure 14 - Accuracy and Loss for a Complex neural Network with 1 Hidden layer of twenty-four neurons ("relu")

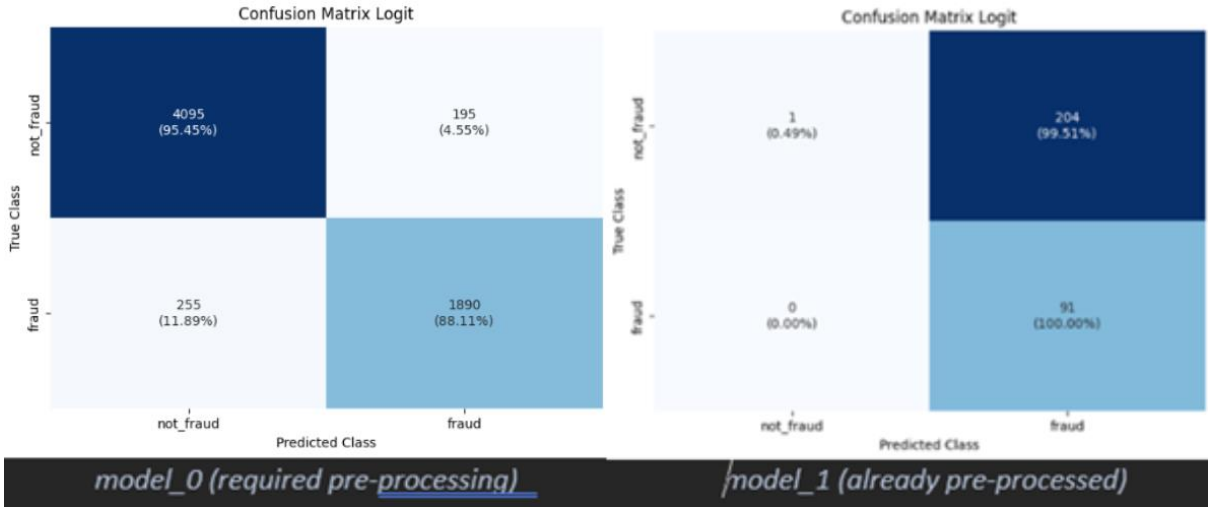

Figure 15 - Confusion Matrix when Hidden Layer of 24 Neurons added ("relu")

7.4.4 Additional Epochs

Model_0 seems to be the one able to satisfy accuracy conditions for a successful ML model to predict fraud with ability to generalize to a wider unseen dataset. However, its learning curves have not yet plateaued after only 10 epochs, therefore additional training time might still improve the results.

```
[62] model_0_c = tf.keras.models.Sequential([
         tf.keras.layers.Dense(24, activation="relu"),
         tf.keras.layers.Dense(24, activation="relu"),
         tf.keras.layers.Dense(1, activation="sigmoid")
     ])

     model_0_c.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                       optimizer=tf.keras.optimizers.Adam(),
                       metrics=["accuracy"])
```

```
import time

start_time = time.time()

history_0_c = model_0_c.fit(X_train_under, y_train_under,
                            callbacks=[helper_functions.create_tensorboard_callback("tensorboard","model_0_c")],
                            epochs=30,
                            validation_data=(X_test_under, y_test_under),
)

end_time = time.time()

total_training_time = end_time - start_time
print(f"Total training timeL {total_training_time:.2f} seconds")
```

```
Saving TensorBoard log files to: tensorboard/model_0_c/20240325-113734
Epoch 1/30
704/704 [==============================] - 6s 5ms/step - loss: 0.3698 - accuracy: 0.8625 - val_loss: 0.3255 - val_accuracy: 0.8864
Epoch 2/30
704/704 [==============================] - 3s 4ms/step - loss: 0.2954 - accuracy: 0.8887 - val_loss: 0.3073 - val_accuracy: 0.8673
Epoch 3/30
704/704 [==============================] - 2s 3ms/step - loss: 0.2741 - accuracy: 0.8909 - val_loss: 0.2873 - val_accuracy: 0.8813
```

Figure 16 - model_0_c architecture and fitting / training process shown.

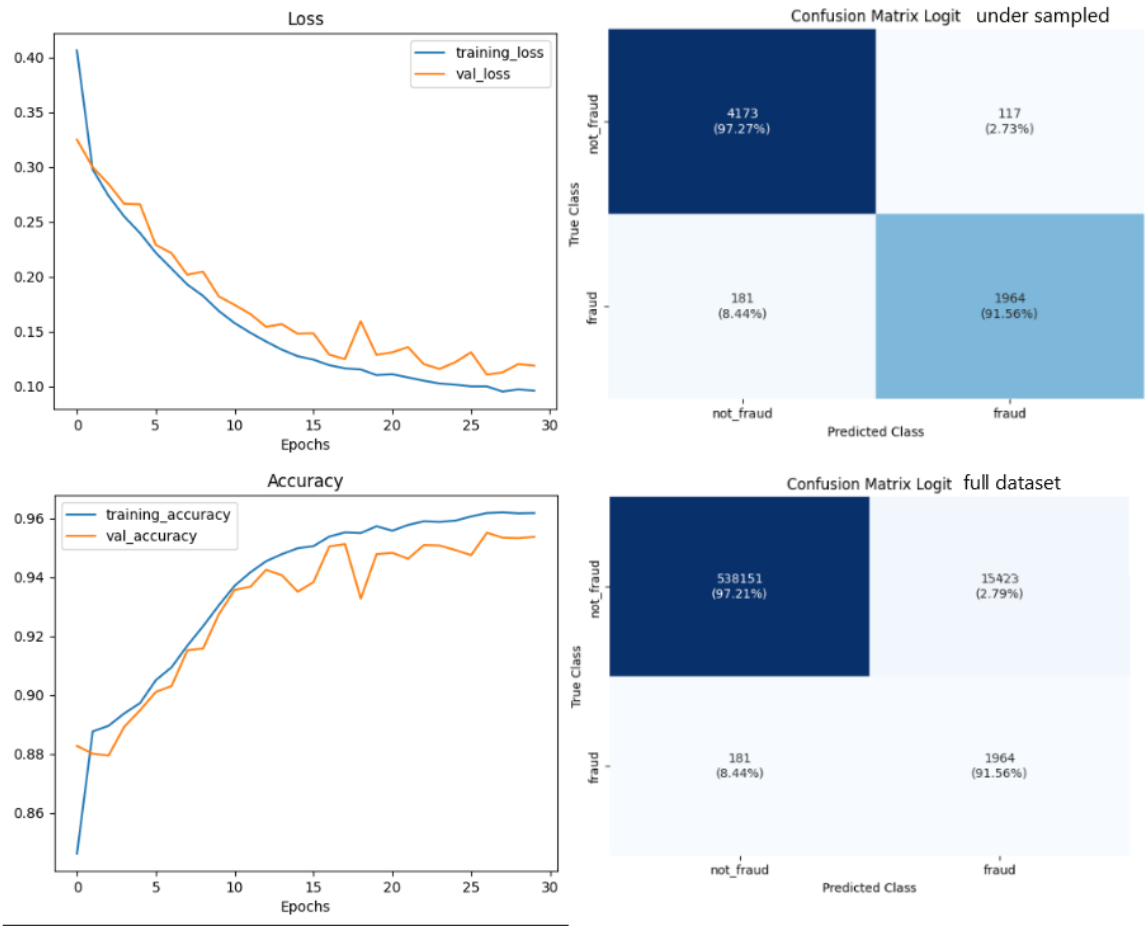

Figure 17 - model_0 (where pre-processing was controlled by us) with Hidden layer of twenty-four neurons ("relu") and given 30 epochs to train.

As can be seen from the figure above, where results of 30 epoch training of our complex neural network are shown, the network is able to predict with 95% success rate against an unseen data, which is a result that should confidently satisfy a business requirement for a fraud detection ML solution.

7.5 Experimentation for Performance

Total training time for the most complex model in this artifact – model_0_c – is a still relatively modest 82.20 seconds for 30 epochs (2-3s per epoch) and considering the size of the dataset GPU processing is not expected to have a significant impact on the training time. In fact, when connected to a T4 GPU on Google Collab the training time for the same model was 155.30 seconds (4-5 seconds per epoch) which is twice as long. This is due to the benefits of GPUs and TPUs being most visible for the ML tasks that require significantly more calculation.

The fraud problem that is being tackled is based around a dataset that is a simple data frame, albeit with over a million rows, however voice and image processing are good examples where the number of calculations grows exponentially.

A typical Convolutional Neural Network employed to process even simplest of images would have many more trainable parameters. For example, one processing a simple 224 by 224 pixel images based

on the efficientnetb0 CNN model has over four million parameters, as compared to our 'complex' neural network model_0_c that has only 2473 parameters.

```
model_0_c.summary()

Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 24)                1848

 dense_6 (Dense)             (None, 24)                600

 dense_7 (Dense)             (None, 1)                 25

=================================================================
Total params: 2473 (9.66 KB)
Trainable params: 2473 (9.66 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model_0.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 224, 224, 3)]     0

 efficientnetb0 (Functional  (None, None, None, 1280   4049571
 )                           )

 global_average_pooling_lay  (None, 1280)              0
 er (GlobalAveragePooling2D
 )

 output_layer (Dense)        (None, 10)                12810

=================================================================
Total params: 4062381 (15.50 MB)
Trainable params: 12810 (50.04 KB)
Non-trainable params: 4049571 (15.45 MB)
_________________________________________________________________
```

Figure 18 - comparison of a simple fraud detection neural network with a single hidden layer with efficientnetb0 model.

7.6 Business Factors and Deployment Approach

Although a successful fraud detection model was developed based on the simulated dataset generated against credit card fraud scenarios, it is not immediately transferable to serve COMPANY's needs.

Firstly, it does not represent COMPANY's business model, which focuses on providing share and pension services. As such the nature of fraud experienced by our company will be different from that found in other parts of the industry. Most importantly, valid features will need to be identified, as we have seen that control of the dataset, its features and pre-processing it is subjected to vastly affect the results achieved and whether the model can generalize to a wider unseen dataset.

Secondly a viable implementation strategy needs to be devised, which would answer the question of where, when, and how the model can act on data in our PROD systems. One option is for it to analyse new transactions live as they are happening and although this approach would provide greatest benefit and ability to proactively tackle fraud it would also face additional complexity in that some features of the transaction are only available after it has been finalized.

Implementation of such a fraud detection model would introduce extra cost and liability for any false positives the model brings up, both in terms of staff time and resources that need to be dedicated to investigating such false positives, but also Service quality and cost due to potential inconvenience for customers should COMPANY decide to act based on a false positive flag from the model. However, it might be argued that the benefit from catching even a single fraudster trying to steal vast sums of money outweighs the necessary operational cost of implementing such a solution.

Furthermore, applying such a model, whether live or after finalization of transactions would require us to fulfil certain legal conditions one of which is acknowledgement from our customers to be assessed by an AI system. Biggest challenge however lies in COMPANY being a data processor for our clients and to implement such a neural network we would need to negotiate access to client PROD data, even if it is only needed for initial anonymization. It could, however, be argued that such data analysis lies within EQs rights to inspect and improve our own services as provided to the client.

The architecture using AWS SageMaker has already been outlined in part one of this report, however it is worth elaborating on the process once the model is exported from the cloud platform. The model

would at that point have been trained and validated on the anonymised dataset(s) and be saved in an exportable / importable format, such as HDF5.

To avoid sending even the anonymized and normalized Production data across to the AWS SageMaker platform which would usually handle deployment through its APIs, COMPANY could setup its own API system on-premises, as the model doing the predictions does not require bespoke ML resources that AWS offers.

Alternatively, COMPANY could embed the model's prediction call directly in relevant application code. This would allow for a low latency solution that can serve as the last gatekeeper before a transaction is finalized. In POC implementation the model could serve as an additional check that fires up a red flag to a relevant human operator, who then investigates the alert. This way the application could operate unchanged, eliminating the risk of the model causing unforeseen delays or cancellations to transactions, which in the case of a live market especially could lead to financial repercussions down the road.

8 PERSONAL REFLECTION ON THE LEARNING EXPERIENCE

This Fraud Detection Neural Network report / proposal is a considerable step-up from the last proposal, which dealt with effectively a third-party solution implemented using either COMPANY's own or cloud architectures. It is also, perfectly aligned with COMPANY's AI policy which does seem to favour solutions developed in-house over those procured from third parties due to data privacy and security constraints. What I have learnt while exploring TensorFlow AWS SageMaker and potential applications of ML / AI within COMPANY will, I have no doubt, help resolve some of the more pressing needs of the company and modernize our processes. The next step for exploring the topic of fraud prevention using complex neural networks at COMPANY, or even any other ML applications developed in house, will be to setup an AWS SageMaker environment that can built upon the network developed in this artifact with a data specifically simulated to fit EQs share and pension transaction scenarios.

What I have learnt about fraud, can be expanded with knowledge from the business about what kind of fraud occurrences we most want to track and predict and a wider collaboration within the business would enable for this proposed application to be fine-tuned to the existing problems within the company.

9 BIBLIOGRAPHY

Amazon Inc. (2024, 03 10). *Amazon SageMaker - Machine Learning Solution*. Retrieved from AWS: https://aws.amazon.com/pm/sagemaker/

Amazon Inc. (2024, 03 10). *Machine Learning Service - Amazon SageMaker Pricing - AWS*. Retrieved from AWS: https://aws.amazon.com/sagemaker/pricing/

Amazon Inc. (2024, 03 10). *Use Amazon SageMaker Built-in Algorithms or Pre-trained Models*. Retrieved from AWS: https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

Change. (2024, 03 11). *PaySim - Change Financial*. Retrieved from Changefinancial: https://changefinancial.com/paysim/

Kulpa, A. (2024, 03 11). *AWS Sage Maker*. Retrieved from MIRO Board: https://miro.com/app/board/uXjVNi7PVuE=/?share_link_id=183189882262

Lopez-Rojas, E. (2024, 03 24). *Synthetic Financial Datasets For Fraud Detection*. Retrieved from kaggle.com: https://www.kaggle.com/datasets/ealaxi/paysim1?rvi=1

Lopez-Rojas, E. A. (2016). Applying Simulation to the Problem of Detecting Financial Fraud. *Blekinge Tekniska Hogskola*.

Nandwani, H., Oyibo, M. E., & Shelton, T. (2022, 01 13). *How Experian uses Amazon SageMaker to Deliver Affordability*. Retrieved from AWS: https://aws.amazon.com/blogs/architecture/how-experian-uses-amazon-sagemaker-to-deliver-affordability-verification/

Shenoy, K. (2024, 03 25). *Credit Card Transactions Fraud Detection Dataset*. Retrieved from Kaggle.com: https://www.kaggle.com/datasets/kartik2112/fraud-detection/data