EXAMINING ANOMALY DETECTION AND REINFORCEMENT LEARNING TECHNIQUES

TERM PROJECT

Group Number: 24

Rebecca Reedel (301454910), Asmita Srivastrava (301436340), Mrinal Goshalia (301478325)

CMPT 318 D1 Fall 2023

Instructor: Uwe Glaesser

ABSTRACT:

This project explores the intersection of energy consumption data analysis and cybersecurity in automated control processes. Analysing key features using feature engineering enhances the understanding of electrical consumption features, contributing to improved system efficiency. Furthermore, effectively deploying Reinforcement Learning to increase efficiency of intrusion detection systems and Hidden Markov Models to assist in anomaly detection, revealing potential cyber threats, including zero-day exploits.

TABLE OF CONTENTS:

TABLE OF FIGURES:

SECTION 1.1 INTRODUCTION

A crucial part of data analysis is understanding the data and underlying features that govern the data distribution. Understanding what the features are and how they contribute to the data can immensely help with feature engineering during the analysis stage. The original dataset has records of energy consumption for years 2006 to 2009, driven by seven response variables namely, Global Active Power, Global Reactive Power, Voltage, Global Intensity, Sub metering 1, Sub metering 2 and Sub metering 3.

It will prove beneficial later on if we take time to understand how these above-mentioned features contribute to our dataset. Global active power represents the real power consumed by electrical appliances for their functioning. On the other hand, Global reactive power is the energy used, not directly for consumption, but for sustenance and maintaining the overall stability of the power system (Damodaran & Vignesh, 2015). Voltage conveys the voltage consumed by the electrical appliances whereas Global Intensity is the total current intensity and provides information about the overall current demand in the system. High intensity levels may indicate periods of increased power consumption. The last three features are about sub metering which can be useful in tracing a more detailed analysis of energy usage in specific areas or for specific purposes (Brownlee, 2019). Therefore, analysing these response variables can help in understanding system performance, and increasing efficiency of electrical consumption.

Throughout this project we will explore cybersecurity techniques for enhancing performance of intrusion detection systems (IDS) through feature engineering and harnessing the capability of probabilistic modelling in Hidden Markov Models (HMMs) to suspect anomalous activities in supervisory control systems like that of energy consumption.

SECTION 1.2 FEATURE ENGINEERING

Feature engineering is a technique used to transform model data by selecting and modelling features variables from raw data in a way to enhance the working of prediction models based on Machine Learning. In cybersecurity practices, feature engineering offers indispensable tools such as feature scaling, dimensionality reduction and encoding of variables to further improve the efficiency of IDS and other anomalous detection models.

1.2.1 Using PCA to determine principal components

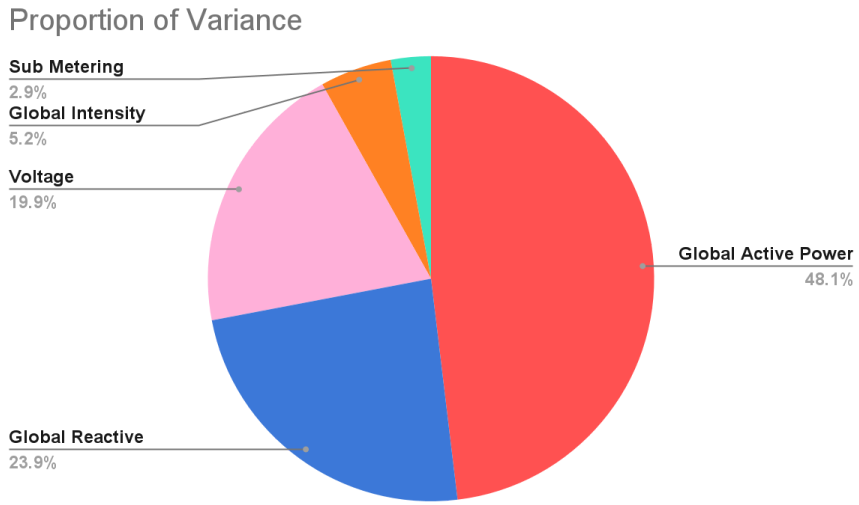

Fig 1: PCA Pie Chart

One of the applications of Principal Component Analysis (PCA) is dimensionality reduction, which is useful in cases where the data is highly redundant and overlapped to visually determine contributing factors (Banerjee, 2022). In such a case, PCA can be a useful feature engineering technique to determine the major contributing features while subduing out the redundancy in data and removing features that contain little to no information about the data under analysis.

In our endeavour to extract the principal components from the seven feature variables, we passed in the scaled dataset for PCA, in return obtaining the summary table as shown in Table 1 and the

plot for the PCA results via the ggbiplot package. From the table, it can be inferred that PC1 - Global_active_power and PC2 = Global_reactive_power contribute majorly to our data about energy consumption. However, while choosing the third principal component, other factors, like assessing variables' correlation to PC1 and PC2 respectively, must be taken into account.

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	9.2381	6.5171	5.9431	3.02488	2.1773	0.7053	0.10669
Proportion of Variance	0.4807	0.2392	0.1989	0.05154	0.0267	0.0028	0.00006
Cumulative Proportion	0.4807	0.7199	0.9189	0.97043	0.9971	0.9999	1.00000

Table 1: PCA results

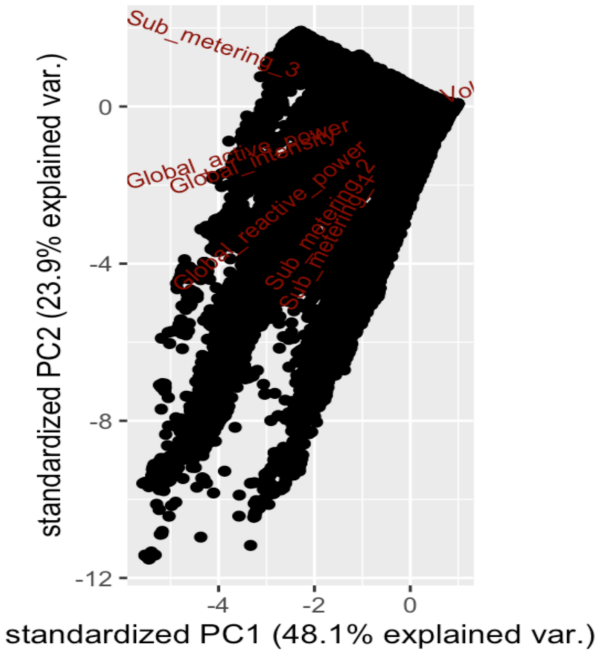

Fig 2: Biplot of PCA results

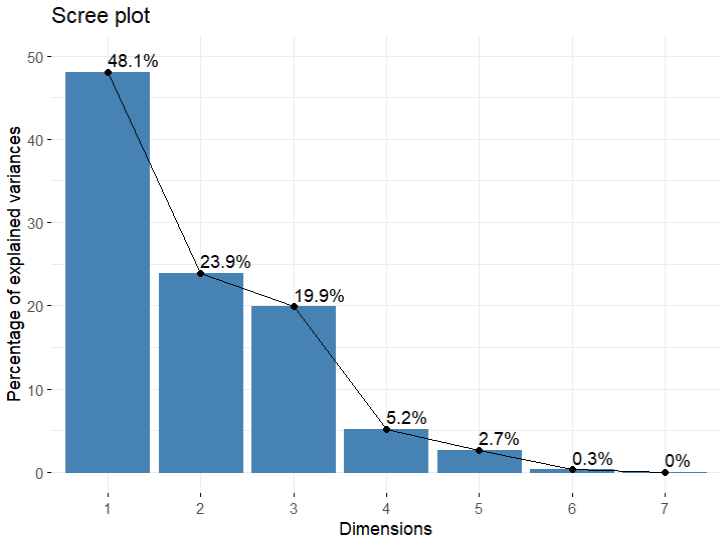

Fig 3: Plot of eigenvalues vs principal components

By using the factor table in Table 2, we can closely assess the contribution of variables while choosing a third principal component for our data assessment. Just by looking at the table 3 below we can choose the third most important feature by its contribution to the principal components. Below is the factoring used to assess the third variable and because Global_intensity influences the PC1(global_active_power) and PC2(global_reactive_power), we consider that as the third most influential variable in the dataset and use these three features to analyse our data.

Response Variables	Contribution to PC1	Contribution to PC2
Global_active_power	4.631402e-01	0.0996963094
Global_reactive_power	3.909151e-04	0.0006986944
Voltage	2.108480e+00	0.5697202583
Global_intensity	1.733467e+01	5.9705662244

Table 2: Factor table for PCA

From Table 2:

Contribution by Voltage : 2.108480e+00*(48.1%) + 0.5697202583*(23.9%)

Contribution by Global_intensity : 1.733467e+01*(48.1%) + 5.9705662244*(23.9%)

Even though Voltage has a higher proportion of variance, its contribution to principal components PC1 and PC2 is lower than Global_intensity. Hence, global intensity contributes more towards the two principal components, thereby making it the third principal component. Therefore, by utilising the technique of PCA, we were able to subset the key response variables that contribute majorly to the data distribution and used Global Active Power, Global Reactive Power and Global Intensity as three variables for training multivariate HMMs on the data about energy consumption.

1.2.2 Rationale behind selection of Response Variables and Time Window

To begin training the Hidden Markov Models, we first needed to obtain a time series of observations for a fixed time window. This was achieved by taking a subset of the scaled data set involving date, weekday, and time values as well as the three principal components as deemed fit above. We then chose a day of the week to reflect the energy consumption pattern for analysis here, by choosing Wednesday. Following this, we accumulated 154 record observations for all Wednesdays from 2006 to 2009 from the dataset provided to us. Choosing the observation time window came as a challenge because we needed to make sure to extract all observations in a given time window. We chose times 12am to 4am on wednesdays because we were able to get **154 wednesdays * 240 minutes worth of observation data = 36960 observations**. For other days and times, we were not able to obtain all observations reflecting a pattern in any way to conduct the HMM model training

SECTION 1.3 HMM TRAINING AND TESTING

After extracting the key features from raw data, one can harness the probabilistic modelling of Hidden Markov Models to train and further identify patterns underlying the data. HMMs can be trained by passing in parameters that speak for the data under assessment, such as n_times being the number of observations of training data and number of states to govern transition between successive state probabilities. HMMs constitute a vital role in the cybersecurity realm to identify anomalous behaviour in data patterns and make IDS more robust.

1.3.1 Comparing Log-Likelihood and BIC Values for Final Model Selection

Our process involved creating an initial 6 models ranging from 4 to 24 states, in increments of 4 each time. Then we compared the Log-Likelihood and BIC difference of each model in order to determine which have the highest LL to lowest BIC ratio. We found by looking for the smallest difference, that the number of states = 20 had the best initial ratio but n_states = 16, 24 were also fairly good so we decided to add a few more models on each side of 20.

Number of States	Log-Likelihood	Bayesian Information Criterion (BIC)	Difference (BIC - Log-Likelihood)
4	-22311.92	44857.54	67169.46
8	-15789.89	32382.49	48172.38
12	-14030.23	29757.3	43787.53
16	-9814.59	22545.32	32359.91
18	-8433.143	20514	28947.143
19	-8940.159	21924.3	30864.459

20	-7754.399	19969.37	27723.769
21	-7577.963	20053.41	27631.373
22	-7561.34	20477.4	28038.74
23	-7124.731	20081.74	27206.471
24	-6609.507	19549.17	26158.677

Table 3: Log-like of all HMMs

Our final selection between the 11 models, ended up being 3 models at n_states = 21, 23, 24. This decision was based on them having the highest log-likelihood to lowest BIC ratio, by comparing the difference between the two values and choosing the smallest total, since a smaller abs(log-likelihood) is preferred over a bigger value since all the LL values were negative.

1.3.2 Explanation of Train and Test Data Splitting

When creating a model we first split the data into a training and a testing set, with a 70/30 proportion respectively. The main driving factor between this choice was to ensure the effectiveness for the model to fit the data. For example, had we not split the dataset, and then after we used the same data with the model, we would have extremely good results, which would mean that we had overfit the data, which essentially just means that the model is just regurgitating what it had previously learned. Which means that if we gave the model, unseen data points it may not be able to perform well and could result in bad results (Sangha, 2021). Conversely, if we were to split the data into train and test, we could use the majority of the data from the initial sample to create the model. Then afterwards we could use the remaining data in the test set, to see how well the model reacts to those new unseen before values, and therefore have a better understanding of the performance of the model (Solawetz, 2022).

1.3.3 Normalised Training vs. Testing Set Results with Final Models

Size of Train Data = 25868, Size of Test Data = 11092

Note: Used Size to normalise results by doing (log-likelihood/size)

Number States	Training Set (Normalised Log-Likelihood)	Testing Set (Normalised Log-Likelihood)
21	-0.29294739	-0.68717166
23	-0.27542643	-0.6198013
24	-0.25550901	-0.60284881

Table 4: Training vs. Testing Set Log-Like

Underfit or Overfit? In order to determine if a model is overfit or underfit we must calculate both the log-likelihoods for the training set and testing set. Afterwards we must normalize them if they are of different sizes, so it is accurate for the number of data points it received. If the resulting training log-likelihood is a lot bigger than the test LL the model is overfitting, vice-versa if the test LL is a lot bigger than the train LL, the model must be underfitting (Brownlee, 2020). We decided that although the training set is bigger than the test set by around a factor of 2, it is fairly close enough that **we do not believe it is overfitting**, and that the previous un-normalized values were almost the same.

SECTION 1.4 ANOMALY DETECTION

Anomaly detection is crucial in all cybersecurity pursuits for detecting any malicious activity over the network by identifying abnormal behaviour indicating security breaches. HMMs can be used to detect anomalous behaviour by specifying a ‘normal’ pattern and using it to identify any deviations and unusual patterns in the data representation.

1.4.1 Comparing Log-likelihood to Determine Anomaly Concentration

In order to determine how anomalous the 3 given data sets were, we used the same number of states for the previous model and created 3 new models, one for each of the data sets. Then, we computed the log-likelihood for each of the models with the dataset values.

Anomalous Data Set Number	Log-Likelihood
1	-14368.92
2	-15192.82
3	-29273.81

Table 5: Log-like of Anomalous Data Sets

As we learned in the class tutorials (R-Tutorials/Part 5), the lower the likelihood (log-like) the less likely the model is to generate the sequence of data given. More specifically to this area of anomaly detection, a sequence of observations that has an **extremely low likelihood**, is because this sequence does not match with the expected behaviour captured by the HMM, and could be more likely an **anomaly**.

Therefore when comparing these anomalous data set log-likelihood results to each other, and knowing that **lower a log-likelihood, means that the dataset is more anomalous.** We can see that the third dataset, which has by far the smallest log-likelihood by a factor of almost x2 compared to the others and therefore, probably contains significantly more anomalies. As well, using that principle, the second dataset should contain the second most amount of anomalies, but still not nearly as much as the third. That said, by looking at these extremely small log-likelihoods, all three datasets are not that well fit by the model. Therefore, there is most likely a notable amount of anomalies in all three of the datasets.

SECTION 1.5 CONCLUSION

Anomaly detection is a vital intrusion detection technique, and is essential to counteract the growing cyber threat landscape of zero-day exploits. This project set out to identify anomalous data through the utilisation of Hidden Markov Models (HMMs) and the evaluation of their log-likelihood. First, after standardisation of the data, the most impactful components were determined using Principal Component Analysis. Next, the remaining data set was split into a train and a test set, this is crucial to ensure our model would react well to unseen data.

From the 11 HMMs developed using the training set, three finalists emerged based on their high log-likelihood and low Bayesian Information Criterion. These finalists underwent further scrutiny by applying a forward-backward technique to ascertain their log-likelihood using the test data. Ultimately, the comparison of log-likelihood values from both the training and test sets led to the selection of a final model with n_states = 24. This chosen model was then applied to three feature-engineered anomalous datasets, and their log-likelihood values were extracted and compared. Following established practices, the lowest log-likelihood is indicative of the presence of anomalies. Notably, Dataset 3 exhibited the lowest log-likelihood, suggesting a higher likelihood of anomalies within that dataset.

Through the integration of Anomaly Detection Techniques involving HMMs and principles of Feature Engineering, this project successfully unveiled potential anomalies within the given datasets, contributing to a more robust understanding and defence against cybersecurity threats.

SECTION 2.1 REINFORCEMENT LEARNING: INTRODUCTION

Reinforcement learning uses the Q-learning algorithm to provide 'memory' to our agent. It enables the agent to learn from the environment's rewards overtime and determine the optimal action to take *(action providing the highest reward)* for a given state. Each state-action pair computes a 'q-value' which represents the 'quality' of an action given a particular state. Better q-values imply greater probabilities of getting higher rewards

SECTION 2.2 INFLUENCE OF HYPERPARAMETERS IN TRAINING PROCESS

Q-learning and thereby the reinforcement model is influenced by three crucial hyperparameters: alpha (α), gamma (γ), and epsilon (ϵ), each falling within the range of 0 to 1. Notably, varying these parameters impacted the q-values, although the reward remains constant regardless.

Alpha (α) represents the model's learning rate. It dictates the degree to which q-values are updated during each iteration. If alpha is set to a low value such as zero, the q-values remain static, resulting in no learning. Conversely, a higher alpha value such as 0.9, ensures frequent q-value updating, facilitating rapid learning. A higher learning rate prioritises recent information rather than previous values and allows the agent to quickly adapt to the environment. However this can make the model more vulnerable to noisy/irrelevant information. A lower learning rate is more conservative, prioritising existing knowledge over recent values. While this approach leads to slower adaptation, it provides stability and accuracy in learning as it is less sensitive to short-term fluctuations/outlying q-values. In our model, we have opted for a moderate alpha value of 0.2. This choice provides a moderate learning rate that relies more on existing knowledge for stability.

Gamma (γ) is the 'discount factor,' determining the weight assigned to future rewards compared to immediate rewards. A lower value such as zero makes the agent solely prioritise current rewards (greedy behaviour). Higher values such as 1 cause the agent to optimise for greater rewards over the long-run. Having very high gamma values causes rewards to be summed up without discounting. Although this can lead to high q-values, the learning process can become increasingly sensitive to variations and noise.This can make it hard for the algorithm to converge, and reach a stable, optimal solution. In our model, we have opted for a balanced gamma value of 0.6, prioritising future rewards slightly more than current rewards as they do provide higher q-values and are generated from many past q-values. Although we do not want to overemphasise them and hence also take into account current rewards to ensure a balanced gamma value for optimal learning.

Epsilon (ϵ) defines the exploration process in the greedy-action selection procedure. In this process, agents explore the environment by selecting an action randomly with probability ϵ. The agent can also use current knowledge by choosing the optimal action with probability 1 - ϵ. Normally, during the action-selection step, agents choose an action based on past q-values. Epsilon and forces the agent to try different actions by picking an action at random. Epsilon enables the algorithm to explore different and potentially better action-selections. This allows it to discover a global optimum and prevents it from being stuck with a local, suboptimal solution. Lower values such as zero cause the agent to never explore and only exploit current knowledge while higher values such as 1 always take random actions and never consider past knowledge. For our mode, we chose a low epsilon value of 0.2. This ensures that the model remains open to

discovering potentially better solutions while staying true to the purpose of the reinforcement model, and making decisions by learning from its existing knowledge.

Epsilon (ϵ) defines the exploration process in the greedy-action selection procedure. In this process, agents explore the environment by selecting an action randomly with probability ϵ. The agent can also use current knowledge by choosing the optimal action with probability 1 - ϵ. Normally, during the action-selection step, agents choose an action based on past q-values. Epsilon and forces the agent to try different actions by picking an action at random. Epsilon enables the algorithm to explore different and potentially better action-selections. This allows it to discover a global optimum and prevents it from being stuck with a local, suboptimal solution. Lower values such as zero cause the agent to never explore and only exploit current knowledge while higher values such as 1 always take random actions and never consider past knowledge. For our mode, we chose a low epsilon value of 0.2. This ensures that the model remains open to discovering potentially better solutions while staying true to the purpose of the reinforcement model, and making decisions by learning from its existing knowledge.

SECTION 2.3 INTERPRETATION OF THE Q-TABLE

The Q-Table stores the q-values for all possible state-action pairs. The states represent the investment values and range from $1-$100 million. The actions represent the investment sectors: Forex, Commodities, Stocks, Real-Estate, and Cryptocurrency. Hence the table displays the expected cumulative reward for each investment.

Based on the parameters chosen we see that the reward is almost always positive. We also find that throughout the states, Commodities and Real-Estate exhibit the widest ranges, indicating

higher variabilities, they also average the highest q-values *(4.7035 and 5.0624 respectively)*. Hence the model indicates that they are the most promising investment sectors. Forex and Stock values have moderate to high q-values and are generally in the mid-range with moderate variability, also being good investment sectors. Cryptocurrencies have both positive and negative q-values. Cryptocurrencies and stocks tend to average the lowest q-values *(4.3917 and 4.2221 respectively).* This coupled with cryptocurrency's variances may indicate it as a higher risk investment category.

Fig 4: Q-table for RL model

State-Action function Q

	Forex	Commodities	Stocks	Real_Estates	Cryptocurrencies
48	1.2098956	4.5568806	4.5446150	4.400272	2.8154534
49	5.9029601	3.9242531	4.6009175	3.555458	3.7463099
50	4.5161098	4.5038675	4.8384741	3.605067	5.4703111
51	4.3097867	4.2233749	5.6307636	3.345917	5.3781451
52	2.3850119	2.9843784	2.9834300	3.996543	3.6323833
53	5.9509997	3.5587119	4.9854958	2.983446	5.0418964
54	3.2604921	5.9837348	5.3643100	3.490358	3.7350383
55	5.4602660	4.1177044	3.7289224	2.171487	4.8961826
56	5.8006126	5.6685604	2.3321259	3.920477	3.5088092
57	3.9282202	5.7470915	3.8887205	3.134019	6.3600439
58	3.1576773	4.5893064	5.0642242	4.990248	5.3828267
59	3.1454083	6.0551787	5.2085820	4.930185	6.0562541
60	4.4371364	6.2415317	4.9817483	5.070882	5.0902753
61	5.5314633	3.1599413	4.1501026	4.845893	5.6355030
62	5.7224735	5.9280122	2.7332510	5.777706	3.0304618
63	3.2639367	4.4525167	4.2664561	5.922276	4.4084239
64	5.9113214	5.8303016	3.5354346	6.016765	4.4401134
65	5.0344926	5.5534581	3.9531637	5.617438	6.0271426
66	2.8195438	5.4641089	5.5736009	3.294869	4.5925979
67	6.1265589	3.3558173	6.2991477	3.842132	5.2556037
68	5.9905716	0.9523774	1.5537884	5.111410	4.4739216
69	4.8706413	6.0627373	3.9886830	5.822843	5.6432569
100	4.7427369	5.1251841	4.3760864	3.221661	2.8171663
70	5.9203156	5.9608799	4.3601050	6.191275	6.0223498
71	4.7670414	5.3267119	4.6609164	4.870117	3.3127511
72	5.4119732	5.1180778	3.3397049	5.728562	4.6041218
73	5.4162009	5.8393638	5.3891095	6.113237	6.5830670
74	5.1971557	5.6861338	5.1493659	4.756417	3.7776405
75	5.0538898	3.2391021	6.6073869	4.414473	3.5437503
76	4.8220497	4.2888783	6.0039609	6.362054	4.4438131
77	6.3652169	3.1355236	4.3618039	6.435424	6.8676358
78	3.1551867	5.0507671	5.4122653	4.094299	4.7168642
1	-0.5933696	3.6663217	-0.5338737	-1.684564	1.9706598
79	6.5913541	6.0428828	3.7567587	2.585123	3.3058424
2	1.2191299	5.2364480	1.2659223	7.405023	0.4643212
3	3.7660030	1.9411512	3.6853096	3.737631	4.5381297
4	4.2341602	2.0701977	4.0209119	5.976501	3.7221887
5	4.0597794	4.5915942	5.2387683	9.519898	5.6921018
6	5.4781812	4.4714823	2.8455636	0.100502	-0.6560000
7	4.5005178	8.3611461	4.5331943	9.200551	7.7930900
10	0.6071290	13.1344486	5.6106023	4.474325	6.2909772
8	5.5589044	5.5640978	4.3123988	6.066382	4.4573639
80	4.4701090	5.6638334	5.6791265	4.993707	5.6524195
11	3.9722732	10.1613071	6.1934667	5.569666	6.3807498
9	2.8956098	3.6718628	5.0654524	2.545518	13.1391812
81	6.2596607	5.3061693	5.4408055	6.776848	6.1836299
12	5.4681843	4.9766902	4.0500279	15.229121	2.6573275
82	2.2728490	3.7286292	4.6218052	6.061959	5.0470611
13	4.4243944	5.0959423	4.4049725	7.439837	3.8998779
83	5.0667234	5.3876487	4.2282057	1.348842	5.2631834
14	6.2597211	7.3686249	4.9998528	11.230141	5.2179695
84	4.2501553	2.2411853	6.4333467	5.611067	5.9051224
15	4.8559124	4.7402151	7.2990009	15.845775	4.5842038
85	5.9264153	4.8350367	6.3727376	3.616594	3.8086442
16	6.0320472	3.8968878	5.6214438	14.265397	4.1213890
86	6.3370733	4.4145631	4.2076461	4.606466	4.9083165
17	5.6208561	4.1730694	5.1466995	12.875819	3.1056416
87	4.4137669	4.5217505	2.1960215	3.499450	3.9774423
18	4.2080111	5.1772954	4.8328071	3.096098	4.8886356
88	2.3043317	4.0869437	6.5814748	2.454090	5.4773759
19	4.5662759	5.3938763	6.7669996	2.303871	3.1889395
89	4.9718013	5.0668569	4.0260795	3.721887	6.3891041
20	3.8426251	5.1450389	5.8684523	6.816137	4.3477071
90	3.2481587	4.7619555	6.6390058	2.735279	4.8811241
21	4.1870112	5.5379334	3.7132382	3.360094	5.3695293
91	4.9397775	4.7603924	5.9879718	2.270694	2.0368640
22	6.5754539	6.1748293	4.3083024	6.628875	3.9787646
92	5.1127711	2.2083443	1.9972866	4.948168	3.6339660
23	1.0548795	4.9234697	5.4328616	3.791269	7.0348245
93	4.1496061	3.8310223	5.0113828	4.188206	5.9957382
24	4.0623869	4.5230631	1.1113458	4.301989	2.1226001
94	3.2674347	5.5095462	5.1322552	5.732456	5.8497018
25	4.5561689	2.8982159	3.0416489	4.858862	3.7903660
95	4.7560837	5.2847972	4.0424263	4.109987	5.2161996
26	4.2208373	4.0833567	4.3975058	6.425837	3.7827690
96	6.2271594	4.6404723	6.1798756	5.071668	4.4929949
27	4.2804260	4.9059847	2.7996259	3.875556	4.4700431
97	5.6208030	4.3461864	5.1579749	4.794983	5.3178750
28	4.6158081	4.6386195	2.8130184	4.214711	4.1534422
98	3.4150530	2.7627356	4.7846345	3.319510	3.6273626
29	2.2087139	3.2614190	3.8215905	4.552846	3.1747474
99	5.7187527	2.2652260	4.0542596	4.224257	4.6470995
30	3.0882944	5.3615043	4.2524331	1.523187	3.3457289
31	4.5592753	3.3583713	3.3283502	4.975918	3.7972025
32	5.2270621	3.8891222	5.1191498	4.745920	2.7061092
33	3.4842937	3.9107923	5.3073450	1.870904	3.4501919
34	4.8949141	4.8903954	4.4469961	3.409544	1.5045871
35	3.4629959	5.4043669	2.1692642	2.799056	4.3014452
36	2.8575624	4.5597445	3.8919734	1.597463	4.2148807
37	2.8337442	2.7374970	5.3344024	4.583496	3.7519052
38	4.6076123	2.5228658	4.5574427	4.266797	4.4081040
39	2.7193127	4.6966070	4.1392674	5.637349	3.3645745
40	3.1997699	5.0253068	4.6559615	5.547474	2.6478099
41	4.9776802	3.4541144	2.8599681	5.086044	6.2440867
42	5.3034296	4.8995688	3.2062067	5.848565	0.9574194
43	5.0029904	5.1354337	0.9896033	4.428420	3.7326405
44	4.8613249	3.9396653	3.5512878	4.948996	3.0679377
45	5.1238183	1.0981249	3.9392909	5.268619	5.2728417
46	5.3421004	5.5842260	4.7501315	4.814659	3.1301041
47	3.2136184	2.6226471	5.7636987	3.554037	5.5481917

SECTION 2.4 POLICY INTERPRETATION

The policy below displays the optimal action to take at each state, maximising expected cumulative reward. This has been computed by comparing and choosing the action with the highest q-value at each state. Since there are 100 states, each state has been assigned its optimal action/investment sector. This could be very useful when determining which sector to invest in given a specific budget. This policy also produces a high, positive reward of 12033.83

suggesting that the investment/trading strategy is successful and produces a significant financial gain.

```
> computePolicy(model)
            48             49             50             51             52             53             54             55
"Commodities"        "Forex" "Cryptocurrencies"       "Stocks"  "Real_Estates"        "Forex"  "Commodities"        "Forex"
            56             57             58             59             60             61             62             63
     "Forex" "Cryptocurrencies" "Cryptocurrencies" "Cryptocurrencies"   "Commodities" "Cryptocurrencies"  "Commodities"  "Real_Estates"
            64             65             66             67             68             69            100             70
"Real_Estates" "Cryptocurrencies"       "Stocks"       "Stocks"        "Forex"   "Commodities"  "Commodities"  "Real_Estates"
            71             72             73             74             75             76             77             78
 "Commodities"  "Real_Estates" "Cryptocurrencies"   "Commodities"       "Stocks"  "Real_Estates" "Cryptocurrencies"       "Stocks"
             1             79              2              3              4              5              6              7
 "Commodities"        "Forex"  "Real_Estates" "Cryptocurrencies"  "Real_Estates"  "Real_Estates"        "Forex"  "Real_Estates"
            10              8             80             11              9             81             12             82
 "Commodities"  "Real_Estates"       "Stocks"   "Commodities" "Cryptocurrencies"  "Real_Estates"  "Real_Estates"  "Real_Estates"
            13             83             14             84             15             85             16             86
"Real_Estates"   "Commodities"  "Real_Estates"       "Stocks"  "Real_Estates"       "Stocks"  "Real_Estates"        "Forex"
            17             87             18             88             19             89             20             90
"Real_Estates"   "Commodities"   "Commodities"       "Stocks"       "Stocks" "Cryptocurrencies"  "Real_Estates"       "Stocks"
            21             91             22             92             23             93             24             94
 "Commodities"       "Stocks"  "Real_Estates"        "Forex" "Cryptocurrencies" "Cryptocurrencies"   "Commodities" "Cryptocurrencies"
            25             95             26             96             27             97             28             98
"Real_Estates"   "Commodities"  "Real_Estates"        "Forex"   "Commodities"        "Forex"  "Commodities"       "Stocks"
            29             99             30             31             32             33             34             35
"Real_Estates"        "Forex"   "Commodities"  "Real_Estates"        "Forex"       "Stocks"        "Forex"   "Commodities"
            36             37             38             39             40             41             42             43
 "Commodities"       "Stocks"        "Forex"  "Real_Estates"  "Real_Estates" "Cryptocurrencies"  "Real_Estates"   "Commodities"
            44             45             46             47
"Real_Estates" "Cryptocurrencies"   "Commodities"       "Stocks"
```

Fig 5: Policy Result Analysis for RL model

SECTION 2.5 OPTIMAL ACTION TABLE FOR EACH STATE

The reinforcement model has been applied to the unseen data with the given budget range of $15-$45 million. We see that real-estate is a very lucrative investment sector, being the optimal action for the most number of states. Commodities also appear to do well in select states and are distributed across the state range. Cryptocurrency and Forex are less lucrative and only appear to do well in a select few states, this indicates that one should proceed with caution when investing in cryptocurrencies.

Fig 6: Optimal Action table for unseen data

State	OptimalAction		
15	Real_Estates	30	Commodities
16	Real_Estates	31	Real_Estates
17	Real_Estates	32	Forex
18	Commodities	33	Stocks
19	Stocks	34	Forex
20	Real_Estates	35	Commodities
21	Commodities	36	Commodities
22	Real_Estates	37	Stocks
23	Cryptocurrencies	38	Forex
24	Commodities	39	Real_Estates
25	Real_Estates	40	Real_Estates
26	Real_Estates	41	Cryptocurrencies
27	Commodities	42	Real_Estates
28	Commodities	43	Commodities
29	Real_Estates	44	Real_Estates
		45	Cryptocurrencies

REFERENCES

Baeldung. (2020, December 18). *Epsilon-Greedy Q-learning | Baeldung on Computer Science*. www.baeldung.com. https://www.baeldung.com/cs/epsilon-greedy-q-learning

Banerjee, A. (2022, December 6). Principal Component Analysis (PCA) in Feature Engineering. *Geek Culture*. https://medium.com/geekculture/principal-component-analysis-pca-in-feature-engineering-472afa39c27d

Brownlee, J. (2020, November 26). *How to identify overfitting machine learning models in SciKit-Learn*. MachineLearningMastery.com. https://machinelearningmastery.com/overfitting-machine-learning-models/

Brownlee, J. (2019, August 5). *How to Load and Explore Household Electricity Usage Data.* MachineLearningMastery.Com. https://machinelearningmastery.com/how-to-load-and-explore-household-electricity-usage-data/

Damodaran, R., & Vignesh, H. S. (2015, October 18). *Analysis of Individual Electricity Usage*. https://rstudio-pubs-static.s3.amazonaws.com/118239_85f5e914687243c69ff97ebaeed6b229.html

Pröllochs, N. (2020, March 2). *Reinforcement Learning in R*. Reinforcement Learning in R. https://cran.r-project.org/web/packages/ReinforcementLearning/vignettes/ReinforcementLearning.html

Sangha, K. (2021, March). *Why do we split the data into train and test sets?* https://karansangha.com/posts/train-vs-test-split

Solawetz, J. (2022, November 18). *Train, validation, test split and why you need it*. Roboflow Blog. https://blog.roboflow.com/train-test-split/

Reinforcement Q-Learning from Scratch in Python with OpenAI Gym. (n.d.). Reinforcement Q-Learning From Scratch in Python With OpenAI Gym – LearnDataSci. https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/