University of
KwaZulu-Natal

June 2022

COMP703 Report: Sequence Modelling Project

Stalk Market Turnip Price Classifier

Prepared by | Sashen Moodley (219006946)

TABLE OF CONTENTS

1 INTRODUCTION

Sequence modelling is a supervised machine learning approach in which a sequence is taken in as input or returned by an appropriate model [1]. This input and output data can take many forms, such as textual, audio, video, or time series, to name a few. Its main differentiating factor from other machine learning paradigms is its focus on dependencies between observations.

Sequence modelling has seen fruitful use in various applications such as speech recognition, music generation, sentiment classification, machine translation, and time series prediction [1]. However, an area that is not widely studied is the upcoming time series classification variant seen in artificial game markets. With various potential candidate artificial game markets to analyse, I was interested in implementing a time series classification for a personal favourite - Nintendo's Animal Crossing: New Horizons (ACNH).

In hopes of exploiting the benefits of sequence modelling, this project aims to determine the efficacy of various machine learning and sequence modelling techniques in classifying an island's turnip prices to its correct weekly trend. Additionally, we hope to provide this reliable trend classification as early in the week as possible.

This approach will come in four broad phases: data acquisition and pre-processing, exploratory data analysis, clustering analysis, and model evaluations.

This report is structured as follows: *Section 2* provides a background into artificial game markets and describes the specific behaviour of ACNH's in-game stock market variant. *Section 3* walks through the data acquisition process. This is followed by *Section 4*, which highlights the data pre-processing activities and provides the results of exploratory data analysis. *Section 5* evaluates two clustering methods (K-Means and DBSCAN) in determining more descriptive trend families. *Section 6* described the experimental setup through which compare and contrast three classifier models (Naïve Bayes, LSTM and GRU). The results of these comparisons are presented in *Section 7*. Finally, these results are discussed in *Section 8*.

2 BACKGROUND

Many games feature an in-game trading space, both online and offline. Games such as Counterstrike: Global Offensive, Eve Online, Warframe, and Elder Scrolls Online instantly spring to mind, but there are tens or potentially hundreds of other games that feature artificial market spaces. Along with an online and offline aspect, game markets have two main approaches when handling currency:

1. *Real-life money*: USD is the most commonly adopted currency with games like CS: GO and
2. *Artificial game-specific tokens/currency*: For example, Warframe uses an artificial currency called platinum for its trading.

With various potential candidate artificial game markets to analyse, I am interested in implementing a time series classification for a personal favourite of mine: Nintendo's *Animal Crossing: New Horizons* (ACNH). This is an offline game featuring an artificial in-game token called *Bells*. Unlike the previous game titles referenced above, *Animal Crossing* is an offline game, thus removing the negative public conception of online trading in video games. It is a light-hearted and child-friendly title that features an in-game market space where the player interacts with the game AI instead of real-life players.

The in-game market is aptly named the *Stalk Market*, as players trade the in-game currency, Bells, in exchange for turnips. The *Stalk Market* follows quite simple mechanics: on Sunday mornings, a turnip vendor named Daisy Mae will visit the player's island, where she will offer to sell her turnips at a randomly generated price. Then during the subsequent days of the week (Monday to Saturday), the island shop will offer to buy the turnips for a price, in *Bells*, that changes twice daily. The aim is to maximise profits by buying low and selling high.

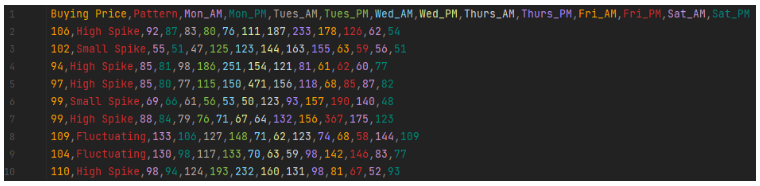
The strategy comes in determining how the turnip prices will unfold (i.e., the trend) as the week progresses. In this project, we will be exploring the weekly trends that the prices follow, with the objective of being able to classify which trend the turnip prices follow based on the first few days of pricing data. We explore classifying variable sequence lengths indicative of how a player would navigate through the week, noting the turnip prices. This trend identification can allow the player to determine when to sell their turnips and thus conquer the *Stalk Market*.

3 DATA ACQUISITION

As with any supervised learning approach, the dataset plays a vital role. As Animal Crossing: New Horizons is a mostly offline game, the availability of public datasets was of particular concern. In the last year there have been various community initiatives to centrally collect players' island turnip prices. Whilst this is an admirable effort, much of this data is characterised by missing values or potential human input errors.

Thankfully, a community member [2], was able to disassemble the game's executable and discovered that turnip price generation follows a randomly generated seed. Various turnip prices throughout the week were extracted and built into the backend of another community member's website [3] to present these findings.

In order to extract this data, to use in our supervised learning approaches, I constructed a web scraping bot using Python's Selenium package (*provided in the submission*). This automated my Chromium-based browser to gather 10,000 weeks of turnip price data which I loaded into a CSV file. This provided the most accurate data for this project, given that it produced 10,000 perfect weeks compared to prior community methods that produced only a few weeks of imperfect data. The snapshot of this raw, unprocessed, data is provided in *Figure 1*.



```
Buying Price,Pattern,Mon_AM,Mon_PM,Tues_AM,Tues_PM,Wed_AM,Wed_PM,Thurs_AM,Thurs_PM,Fri_AM,Fri_PM,Sat_AM,Sat_PM
106,High Spike,92,87,83,80,76,111,187,233,178,126,62,54
102,Small Spike,55,51,47,125,123,144,163,155,63,59,56,51
94,High Spike,85,81,98,186,251,154,121,81,61,62,60,77
97,High Spike,85,80,77,115,150,471,156,118,68,85,87,82
99,Small Spike,69,66,61,56,53,50,123,93,157,190,140,48
99,High Spike,88,84,79,76,71,67,64,132,156,367,175,123
109,Fluctuating,133,106,127,148,71,62,123,74,68,58,144,109
104,Fluctuating,130,98,117,133,70,63,59,98,142,146,83,77
110,High Spike,98,94,124,193,232,160,131,98,81,67,52,93
```

Figure 1 – Snapshot of raw data from web scraping bot

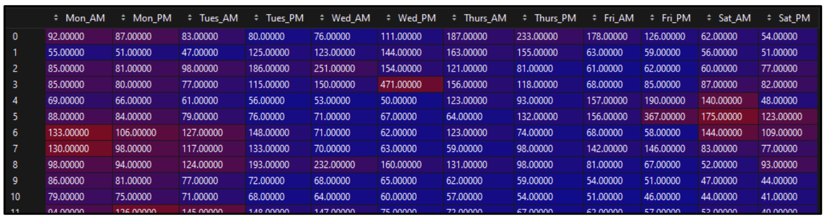
Section 4 explores this in greater detail by going through the process of cleaning, pre-processing, and filtering this raw data to its final form, which will be fed into our various classifier models.

4 DATA PRE-PROCESSING AND EXPLORATORY ANALYSIS

The weekly turnip price data, in the form of a CSV file, is loaded into a *Pandas Dataframe*, where it is appropriately subject to the data processing and analysis activities described in the following sections. Additionally, other popular data processing and analysis packages, such *as NumPy*, *scikit-learn*, *Keras pre-processing*, and *Matplotlib*, are utilised.

4.1 COLUMN FILTERING

Since the franchise's first entry back in 2001, the Animal Crossing community collectively views the possible stalk market trends as: high spike, small spike, decreasing, and random. As such, the owner of the website [3] from which our data is scraped, has labelled the turnip prices following these four trends. Whilst this is appropriate to determine the overall week pattern, it is somewhat restrictive because it does not indicate when this spike can occur. This is the primary motivation for employing clustering techniques, discussed later on in *Section 5*, to detect better trend descriptors for our turnip prices. Consequently, the scraped 'Pattern' column is dropped. As the buying price has no function in classifying this sequence, the 'Buying Price' column is similarly dropped. The prices are also converted to float values.



	Mon_AM	Mon_PM	Tues_AM	Tues_PM	Wed_AM	Wed_PM	Thurs_AM	Thurs_PM	Fri_AM	Fri_PM	Sat_AM	Sat_PM
0	92.00000	87.00000	83.00000	80.00000	76.00000	111.00000	187.00000	233.00000	178.00000	126.00000	62.00000	54.00000
1	55.00000	51.00000	47.00000	125.00000	123.00000	144.00000	163.00000	155.00000	63.00000	59.00000	56.00000	51.00000
2	85.00000	81.00000	98.00000	186.00000	251.00000	154.00000	121.00000	81.00000	61.00000	62.00000	60.00000	77.00000
3	85.00000	80.00000	77.00000	115.00000	150.00000	471.00000	156.00000	118.00000	68.00000	85.00000	87.00000	82.00000
4	69.00000	66.00000	61.00000	56.00000	53.00000	50.00000	123.00000	93.00000	157.00000	190.00000	140.00000	48.00000
5	88.00000	84.00000	79.00000	76.00000	71.00000	67.00000	64.00000	132.00000	156.00000	367.00000	175.00000	123.00000
6	133.00000	106.00000	127.00000	148.00000	71.00000	62.00000	123.00000	74.00000	68.00000	58.00000	144.00000	109.00000
7	130.00000	98.00000	117.00000	133.00000	70.00000	63.00000	59.00000	98.00000	142.00000	146.00000	83.00000	77.00000
8	98.00000	94.00000	124.00000	193.00000	232.00000	160.00000	131.00000	98.00000	81.00000	67.00000	52.00000	93.00000
9	86.00000	81.00000	77.00000	72.00000	68.00000	65.00000	62.00000	59.00000	54.00000	51.00000	47.00000	44.00000
10	79.00000	75.00000	71.00000	68.00000	64.00000	60.00000	57.00000	54.00000	51.00000	46.00000	44.00000	41.00000
11	94.00000	126.00000	145.00000	148.00000	147.00000	75.00000	72.00000	67.00000	62.00000	57.00000	53.00000	49.00000

Figure 2 – Dataframe after filtering

As we can see from the raw data in *Figure 1*, our data does not follow the traditional time series format with the data and price as separate columns. This is because we are not performing forecasting, per se, but we are instead classifying our turnip prices into one of the trend families based on a full week of prices. This format of placing each time value (Mon_AM, Mon_PM, Tues_AM, …, SAT_PM) in its own column will benefit us when we perform our classification and clustering analysis, as rather than working in $\mathbb{R}^2$ we can work in $\mathbb{R}^{12}$. The results of these filtering activities, up to this point, is shown in *Figure 2*.

4.2 DATA PRE-PROCESSING

As we can see in *Figure 2*, our price data assumes various ranges. This will require some scaling. As the spikes are vital features of the weekly price data, it would be unwise to scale the data in [0,1] interval as this will eliminate (squash) the trend characteristic. Therefore, we will attempt to normalise the price data to follow a standard normal distribution (i.e., $\mu=0$ and $\sigma=1$). As indicated by [4], this type of normalisation is very common for time series classification problems. This was achieved via the *StandardScaler* of *scikit-learn's pre-processing* package.

4.3 EXPLORATORY DATA ANALYSIS

Before going into clustering activities, I performed some exploratory data analysis on the unscaled column-filtered data to develop a hypothesis of the expected clustering results. Firstly, a visualisation of the first ten weeks from our turnip dataset is plotted in *Figure 3*.

Figure 3 clearly validates the community's view of the four prominent trend families. We can see that the red line follows a "high spike", whilst the green and blue lines follow a "small spike", and several lines are decreasing or seemingly random.

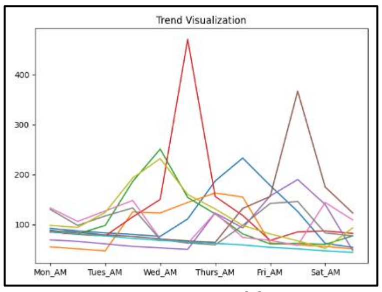
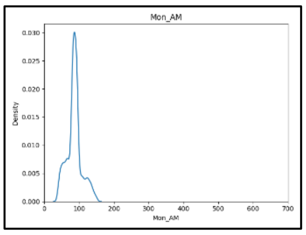
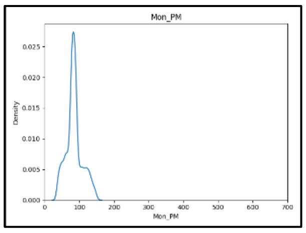
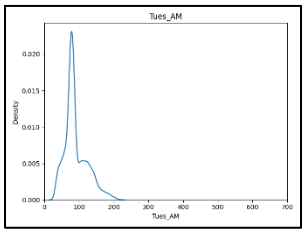
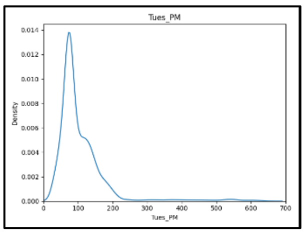
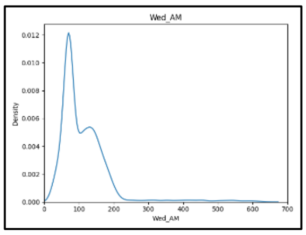
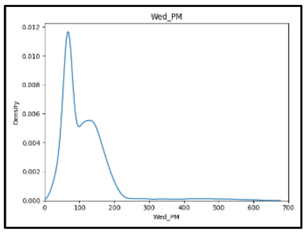
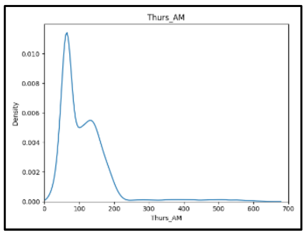
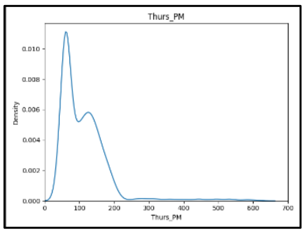
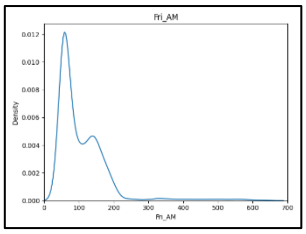
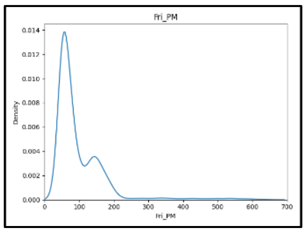
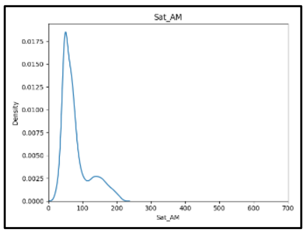
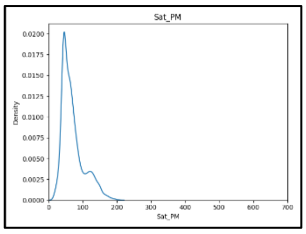


Figure 3 – Visualization of first 10 samples

Additionally, I generated kernel density plots for the daily price distribution represented in *Table 1* below.

Table 1 – Kernel density plots of the daily price distribution

Monday AM	**Monday PM**	**Tuesday AM**	**Tuesday PM**
			
Wednesday AM	**Wednesday PM**	**Thursday AM**	**Thursday PM**
			
Friday AM	**Friday PM**	**Saturday AM**	**Saturday PM**
			

As we can see from *Table 1*, all the days seem to indicate some form of bimodality within the range of 0 – 200 bells. Also, for every day, we can see that the most probable values fall in the range 0-100, with the second most probably range being 100-200 bells. However, notably we can see that from 'Tues_PM' to 'Fri_PM' there is a minuscule probability of values in the range of 300-600 bells. This is indicative of a "high spike", and these kernel plots show how infrequently they occur.

5 CLUSTERING ANALYSIS

Clustering analysis is a machine learning paradigm concerned with grouping a set of observations such that objects in the same group are more similar to each other than those in other groups [5].

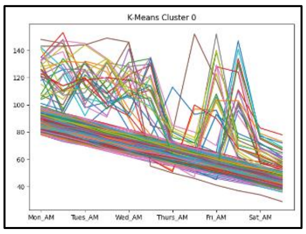
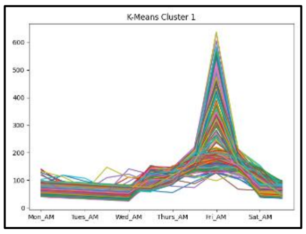
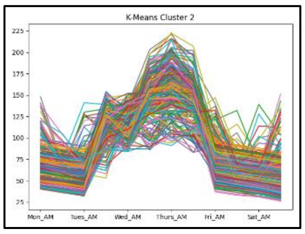
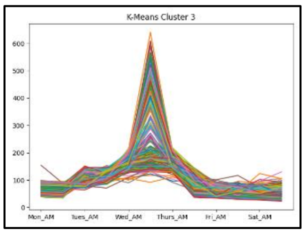
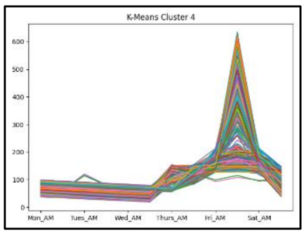
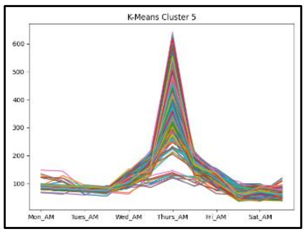
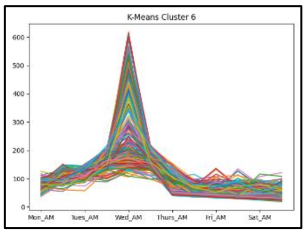
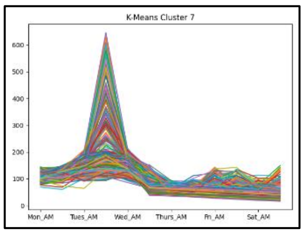
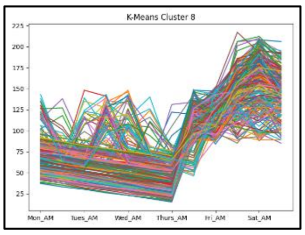
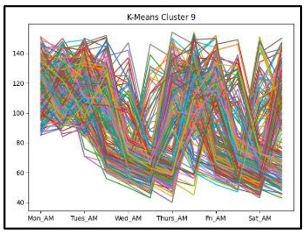
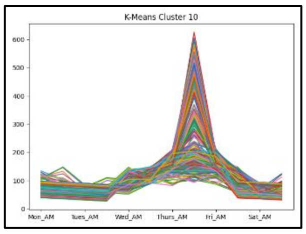
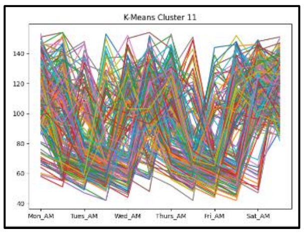
As mentioned in *Section 4*, the aim of the clustering analysis in this research is to extend the community's views of the 4 general trends: high spike, small spike, decreasing, and random. These new turnip price family trends, resulting from clustering, aims to be more descriptive as they can indicate when we could expect the four major trends to occur. As a result, although this is not a forecasting task, we can still express/convey some level of forecasting within our classification class itself.

From the visualisations in *Table 1*, a hypothesis is made of identifying at least 12 different trend classification families, indicative of the four overall trends and, more interestingly, when they can occur during the week.

5.1 K-MEANS

The first clustering technique employed is the K-means clustering. This is implemented via the *KMeans* class of *scikit-learn's cluster* package. Based on my hypothesis, I set the number of clusters to '12', and fit the instance to the scaled turnip prices. The resulting clusters can be visualised through the plots represented in *Table 2* below:

Table 2- K-Means clustering visualisations

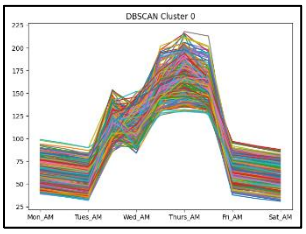
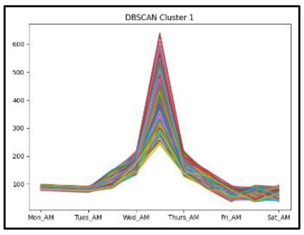
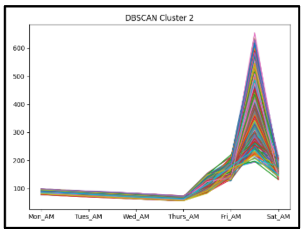
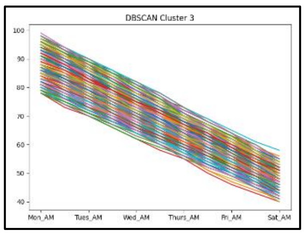
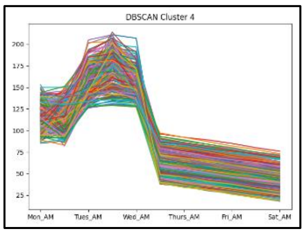
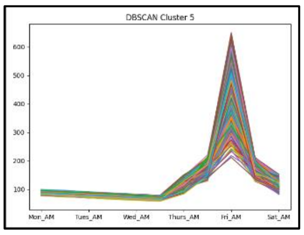
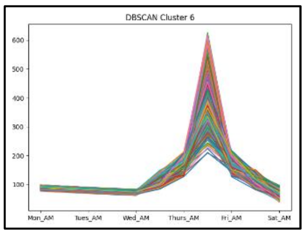
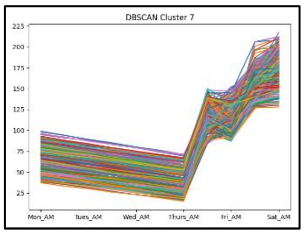
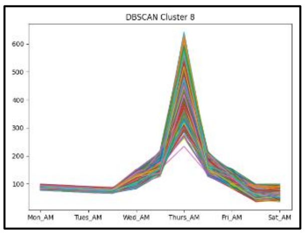
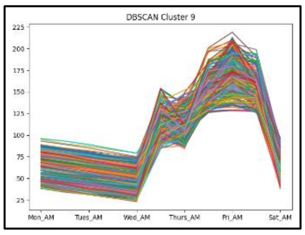
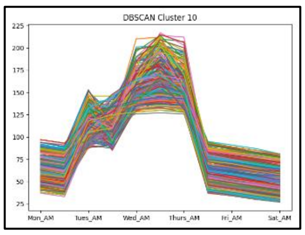
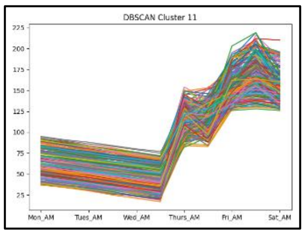
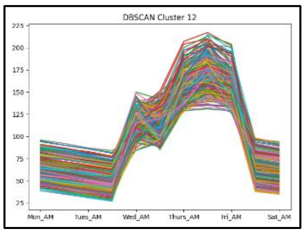
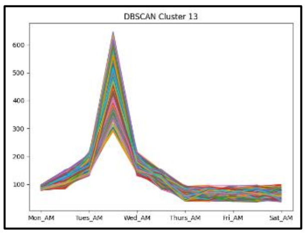
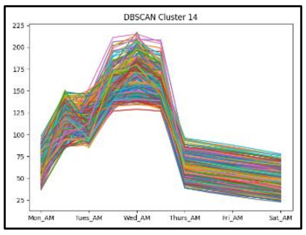
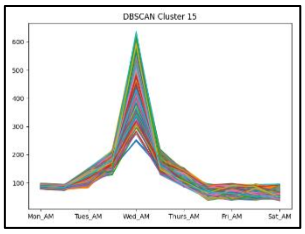
Cluster 0	**Cluster 1**	**Cluster 2**	**Cluster 3**
			
Cluster 4	**Cluster 5**	**Cluster 6**	**Cluster 7**
			
Cluster 8	**Cluster 9**	**Cluster 10**	**Cluster 11**
			

As we can see, K-Means does a decent job at segmenting the trends, but still leaves much to be desired. These results, along with their comparison against DBSCAN, are examined in greater detail in *Section 5.3*.

5.2 DBSCAN

The second clustering technique employed is DBSCAN, which hopes to provide a better clustering of turnip prices than K-means. Similarly, DBSCAN is implemented using the *DBSCAN* class of the *scikit-learn's cluster* package. The minimum samples were set to '18', and the instance was fit to the same scaled turnip prices. The resulting clusters can be visualised through the plots represented in *Table 3* below:

Table 3- DBSCAN clustering visualizations

Cluster 0	**Cluster 1**	**Cluster 2**	**Cluster 3**
			
Cluster 4	**Cluster 5**	**Cluster 6**	**Cluster 7**
			
Cluster 8	**Cluster 9**	**Cluster 10**	**Cluster 11**
			
Cluster 12	**Cluster 13**	**Cluster 14**	**Cluster 15**
			

5.3 CLUSTERING RESULTS DISCUSSION

Our first attempt at clustering with K-means did not yield the most desirable results. We can see that there still exists some noise in certain clusters which were not filtered. For example, if we analyse 'Cluster 8' we can see that we are, for the most part, identifying 'Sat_PM' as a small spike with observed turnip prices peaking at '225' bells. However, there are still line plots denoting the random trend earlier in the week. If anything, K-means provided an indication that we could indeed find clusters within the data to provide further description of the four main trends.

The second clustering technique, DBSCAN, yielded much more desirable results, as each cluster can be uniquely identifiable from one another, with little to no overlapping noise from other clusters. This can be explained by DBSCAN's approach to minimum samples, which as indicated earlier was set to '18' [6]. This parameter will classify any weekly observation that does not have at least '18' neighbours as an outlier, subsequently being placed in 'Cluster -1' represented in *Figure 4*. 'Cluster -1' captures all the outliers (noise) in our data, thus enabling the remaining clusters to be well-segmented by both their main family trend (high spike, small spike, decreasing, and random) and the day and time on which the prices reach their peaks.

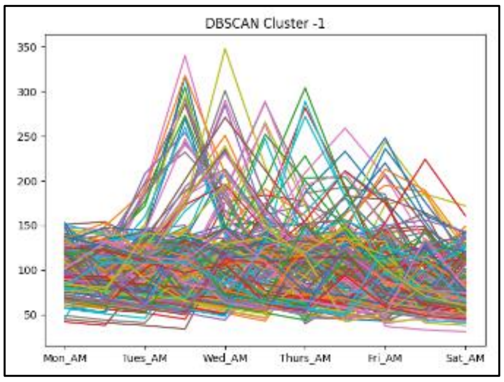
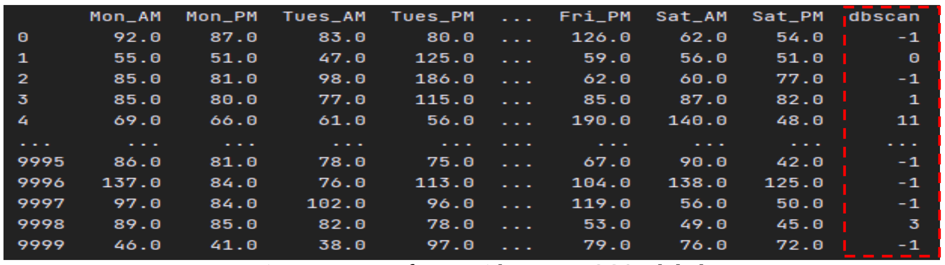


Figure 4 – Cluster '-1' capturing outliers

Consequently, DBSCAN was chosen as the clustering technique of choice, and our weekly turnip prices data frame is extended with their respective DBSCAN clustering family trend class, which acts as our labels. This extended data frame is presented in *Figure 5*.



```
      Mon_AM  Mon_PM  Tues_AM  Tues_PM  ...  Fri_PM  Sat_AM  Sat_PM  dbscan
0       92.0    87.0     83.0     80.0  ...   126.0    62.0    54.0      -1
1       55.0    51.0     47.0    125.0  ...    59.0    56.0    51.0       0
2       85.0    81.0     98.0    186.0  ...    62.0    60.0    77.0      -1
3       85.0    80.0     77.0    115.0  ...    85.0    87.0    82.0       1
4       69.0    66.0     61.0     56.0  ...   190.0   140.0    48.0      11
...      ...     ...      ...      ...  ...     ...     ...     ...     ...
9995    86.0    81.0     78.0     75.0  ...    67.0    90.0    42.0      -1
9996   137.0    84.0     76.0    113.0  ...   104.0   138.0   125.0      -1
9997    97.0    84.0    102.0     96.0  ...   119.0    56.0    50.0      -1
9998    89.0    85.0     82.0     78.0  ...    53.0    49.0    45.0       3
9999    46.0    41.0     38.0     97.0  ...    79.0    76.0    72.0      -1
```

Figure 5 – Dataframe with new DBSCAN labels

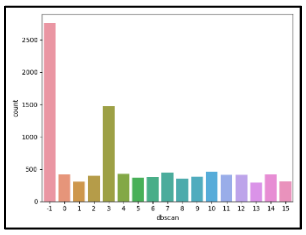
6 EXPERIMENTAL SETUP

Once the clustering analysis is concluded, the final steps in data preparation are made, and the experimentation can begin. Three classifier models will be constructed and compared: A Naïve Bayes, an LSTM, and a GRU. The Naïve Bayes model will be used as a comparative baseline for the Recurrent Neural Networks (RNN) architectures. The experimentation will be performed on an appropriately equipped local HPC device.

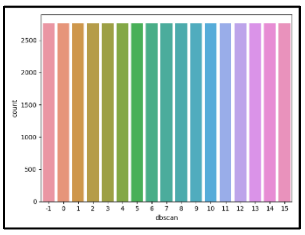
6.1 DATA PREPARATION

Before the models can be constructed, the turnip price data from the clustering results need to be appropriately prepared to be fed into these models. Firstly, to ensure the data is balanced, *imblearn*'s *RandomOverSampler* was utilised to provide an even distribution across the family trends. *Figure 6b* indicates this even distribution. Consequently, the random over-sampling duplicates minority classes to provide a balanced dataset. This increased the number of samples in our original dataset from 10,000 to 46,903 total samples.

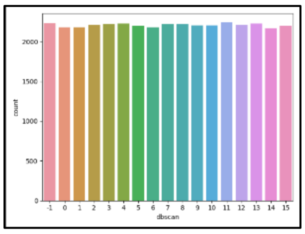
Secondly, an 80:20 split will be made on the data constituting the training and testing set respectively. This split is made using a random state seed courtesy of *sklearn*. The training set was further split in an 80:20 fashion, with 10% of the training data being set aside for validation. This resulted in 30018, 7504, and 9381 weekly turnup price entries for training, validation, and testing respectfully. As indicated by *Figure 6c*, our data remains relatively balanced after the split.



(a) – Before over-sampling or split



(b) – After over-sampling



(c) – After over-sampling and split

Figure 6 - Results of Random Over-Sampling

The final phase in preparing the data was encoding the family trend labels. This was achieved through *sklearn*'s *LabelEncoder* Class and *Keras to_categorical* function.

As indicated in *Section 1* and *2*, we aim to have our models classify the week's trend based on variable input sequence lengths rather than the full 12 prices throughout the week. This is done for practical reasons, because we want our classifier to detect the trend as early in the week as possible for the player to capitalise on bigger profit gains. This segmentation also allows our classification to be representative of how a player would navigate through the week - noting the turnip prices twice every day and dynamically growing their list of prices. For the purposes of the main report, we will consider classifying the first '6' prices of the week. However, we expect the accuracy of the models to increase accordingly as the number of considered points increase and we can see this through the additional testing performed on the first '4', '8', and '10' prices which have been added to *Appendix A*. These results are additionally referenced in *Section 8* when evaluating model performance. Therefore, the final data preparatory step involves selecting the first '6' columns, corresponding to half the weekly prices, to classify into our DBSCAN clustering trend families.

6.2 MODELS

Inspired by another community member's work in creating a Bayesian inference turnip price **forecaster** [7], a Bayesian network-based classifier was created for this project. The Naïve Bayes classifier is chosen to act as a baseline for our two Recurrent Neural Network models (LSTM and GRU). The Naïve Bayes classifier is a simple probabilistic Bayesian network model incorporating strong independency assumptions. It also serves to represent current (non-sequential) attempts to classify turnip prices, which our RNN sequential models would hope to surpass in providing more accurate classifications.

Recurrent Neural Networks (RNNs) are a deep learning and AI Neural Network design which is particularly suited towards sequential data processing. RNNs and its GRU and LSTM variants maintain an internal memory which is useful for capturing sequential input in machine learning. As stated by [8], the key benefit they offer over Convolutional Neural Networks is that RNNs allow characteristic weights to be shared over time. That is, they can recall their prior inputs.

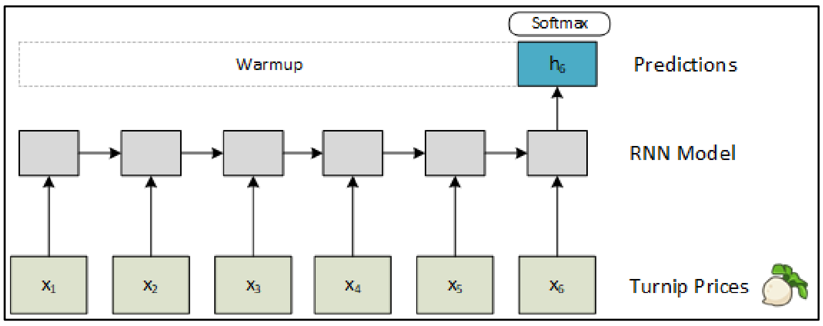


Figure 7- RNN architecture for 6 prices

The overall RNN architecture comes in various flavours: one-to-one, one-to-many, many-to-one, and many-to-many. As this is a sequence classification task with a single output, we will be using the many-to-one architecture. The architecture of our RNN for ‘6’ turnip price (half the week) observations is provided in *Figure 7*.

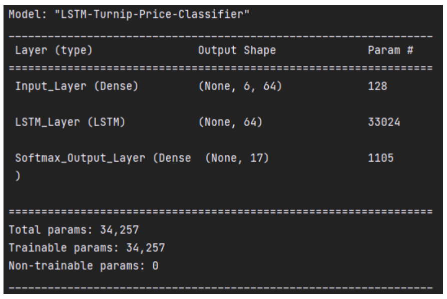
As we can see, our RNN has ‘6’ observational timesteps with one feature per timestep, which is our turnip price. The RNN will update its internal memory and weights with every noted price occurrence. The input data is shaped as a 3-dimensional array to represent the batch size, timesteps, and input dimension respectfully. After the 6th timestep, the RNN will predict the most likely classification.

6.2.1 NAÏVE BAYES

The Naïve Bayes classification model is implemented via *scikit-learn’s GaussianNB* class. The classifier is fit on the pre-processed training data and is provided with the appropriately encoded DBSCAN labels, allowing it to make predictions.

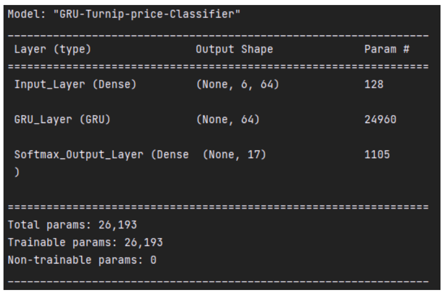
6.2.2 RNN - LSTM AND GRU

Model training for both the LSTM and GRU models is facilitated vai *Keras Sequential* architectures. These architectures are set up in a manner that allows for a consistent comparison of results by only swapping out the LSTM layer for the GRU layer or vice versa. The model architectural summaries for both the LSTM and GRU are provided in *Figure 8* below:



```
Model: "LSTM-Turnip-Price-Classifier"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_Layer (Dense)         (None, 6, 64)             128

 LSTM_Layer (LSTM)           (None, 64)                33024

 Softmax_Output_Layer (Dense  (None, 17)               1105
 )

=================================================================
Total params: 34,257
Trainable params: 34,257
Non-trainable params: 0
_________________________________________________________________
```

(a) – LSTM



```
Model: "GRU-Turnip-price-Classifier"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_Layer (Dense)         (None, 6, 64)             128

 GRU_Layer (GRU)             (None, 64)                24960

 Softmax_Output_Layer (Dense  (None, 17)               1105
 )

=================================================================
Total params: 26,193
Trainable params: 26,193
Non-trainable params: 0
_________________________________________________________________
```

(b) – GRU

Figure 8 – Model architecture summaries

No regularization or dropout layers were required as the training did not suffer from overfitting (see *Appendix B*). Both models are compiled similarly, once again to allow for a fair comparison of results. Models are trained across ‘100’ epochs with a batch size of ‘64’. The loss function used is *Categorical Cross entropy, which* is most suitable for multi-class classification. *Adam* optimisation algorithm is used as it combines the benefit of both *AdaGrad* and *RMSProp* [9]. Additionally, dynamic learning rates, model checkpointing, and early stopping procedures are incorporated into the training via *Keras ReduceLROnPlateau*, *ModelCheckpoint*, and *EarlyStopping callbacks*, respectively. This combination of *callbacks* provides a reduced risk of overfitting, ensures we keep a persistent copy of the best model, and prevents unnecessary computing cycles in the case of a validation accuracy plateau.

6.3 METRICS

The following proves metrics, provided in *Table 4*, prove to be valid candidates in allowing for a fair comparison of these models.

Table 4 – Metrics for model evaluation and comparison

Metric	Description	Formula
Accuracy	Indication of subset accuracy: the set of predicted labels must exactly match the corresponding set of true y labels.	$A = \frac{TP + TN}{TP + TN + FP + FN}$

Precision	Ratio of correctly predicted positive observations to the total predicted positive observations	$P = \frac{TP}{TP + FP}$
Recall	Ratio of correctly predicted positive observations to all observations in actual class	$R = \frac{TP}{TP + FN}$
F1-Score	Weighted average of Precision and Recall	$F1 = 2 \times \frac{P \times R}{P + R}$
Loss (CCE)	A measure on how bad the model's prediction was on a single sample.	$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} 1_{y_i \in C_c}\, log(p_{model}[y_i \in C_c])$
MSE	Measures the amount of error in statistical models (distance between observed and predicted)	$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$

7 RESULTS

With the experimental setup specified in *Section 6*, the following results were gathered for classifying the **first 6 turnip price observations** during the week, which provides the basis of our discussion in Section 8. Note, that the results for the other sequence lengths (4, 8, and 10) are provided in *Appendix A.*

Table 5 – Confusion matrices for classification predictions when given first 6 turnip prices

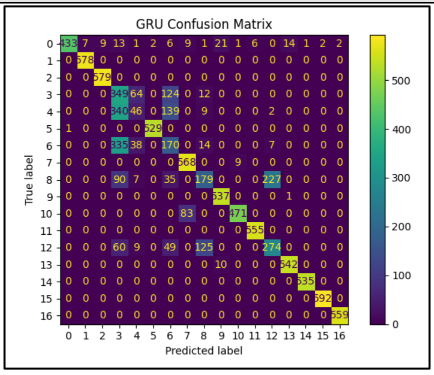
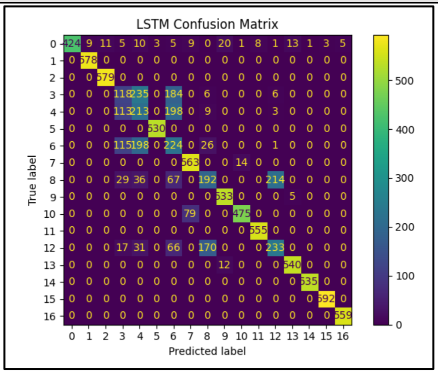
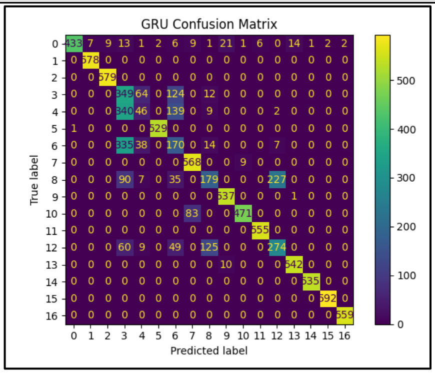
Confusion Matrix		
Naïve Bayes	LSTM	GRU
		

Table 6 – Summary of classification report when given first 6 turnip prices

		Weighted Average Metric		
		Precision	**Recall**	**F1-Score**
Classifier	**Naïve Bayes**	0.76	0.77	0.75
	LSTM	0.80	0.79	0.79
	GRU	0.81	0.80	0.79

Table 7 - Metric scores obtained when evaluating the RNN Architectures on the first 6 turnip prices

		Metric					
		Accuracy	**Loss**	**MSE**	**F1-Score**	**Precision**	**Recall**
RNN Architecture	**LSTM**	0.7934	0.4333	0.0135	0.8164	0.9278	0.7315
	GRU	0.7991	0.4292	0.0134	0.8218	0.9327	0.7370

Table 8 - Elapsed time of each classifier during data fitting, evaluation, and prediction of the first 6 turnip prices

		Stage		
		Data Fitting (s)	**Evaluation (s)**	**Prediction (s)**
Classifier	**Naïve Bayes**	0.007	N/A	0.0030
	LSTM	847.992	3.362	2.704
	GRU	816.662	3.316	2.806

Note: The training and validation metric visualisation plots have been provided in *Appendix B*. Once again, the relevant results for other dynamic sequence lengths (4, 8, and 10) are presented in *Appendix A*.

8 DISCUSSION

From the results presented in *Section 7* and *Appendix A*, the following main observations were made:

1. The LSTM and GRU provide more accurate classifications than the Naïve Bayes classifier.
2. The classification accuracy is directly proportional to the number of provided turnip prices during the week.
3. As the number of provided turnip decrease, the accuracy difference between the naïve classifier and the RNN models increase.
4. The GRU is the best performing classifier.

Diving into this deeper, we can try to understand these observations. Firstly, it is encouraging to see that both RNN architectures (LSTM and GRU) consistently outperform the baseline set by the Naïve Bayes (NB) classifier across all tested sequence lengths of 4, 6, 8, and 10 prices. This is observed in all metric categories presented in *Table 4*, most notably the F1-Score, where the RNNs performed 6% better, on average, than the NB classifier. This difference in accuracy is owing to two main factors. Firstly, the NB and RNN models take different stances on sequential dependencies, or lack thereof. The NB classifier makes the strong independency assumption between the weekly turnip prices. In contrast, the RNNs aim to maintain a dependent memory state of previously observed prices in the sequence. Secondly, is the fact that the NB classifier is a generative model whereas the LSTM and GRU are discriminative models. That is, the Naïve Bayes classifier aims to describe how the data was generated in the first place. It seeks to find the distribution which yielded the input examples. Conversely, the discriminative nature of the RNNs takes a more direct approach to classification in identifying only the most discernible features amongst the different classification classes.

Next, in elaborating on *point 2*, we find that the accuracy of all our classifiers is directly proportional to the length of the turnip price sequence that we allow the models to view when making the classification. Referring to *Tables 6, A-2, A-6,* and *A-10*, we can see the average F1-Score across all classifiers steadily increases from 53% (when given only the first 4 weekly prices) up to 96% and 98% when given the first 8 and 10 prices, respectively. This observation confirms my hypothesis, that as we progress through the week, and make additional price observations, our classifiers should be able to provide a better indication of the potential trend throughout the week. Naturally, this is due to the classifiers having more data points to discern against their DBSCAN-generated labels.

Thirdly, we find that the accuracy differences between the Naïve classifier and the RNNs increase for shorter length sequences. Also, interpreting this the other way, we find that as more prices are noted throughout the week, the accuracy difference between the Naïve Bayes classifier and the RNNs decreases. This can be explained by the kernel density plots shown in *Table 1* and DBSCAN clustering analysis in *Table 3*. We can see that most of the characteristic determinants of these trend families are given in the first half of the week, and once we surpass these 6 price observations, we find diminishing advantages of the RNNs over the Naïve Bayes classifier. However, this is expected and drawing our attention back to the main aim of the classification, which is to determine the price pattern as early in the week as possible, we find this observation to be favourable to the RNNs. Examining why this is the case, we can contribute to *point 1*'s discussion by elaborating on the conditional independency assumption. Applying this to our stock market price classification domain, including artificial market spaces such as this, all the information of the previous price observations is captured in the current price. In fact, the only notable advantage that the Naïve Bayes classifier has over the RNNs is when it comes to data fitting and prediction times (refer to *Tables 8, A-4, A-8,* and *A-12*).

Lastly, in making a single model selection for our turnip price classification task, it is empirically evident, through the results gathered in *Tables 6-8*, that the GRU would be the most suitable choice given its accuracy and speed performance. As we have already examined, the GRU confidently outperforms the accuracy baseline set by the Naïve Bayes classifier across all investigated weekly turnip price observations. Compared to LSTM, the GRU produces slightly more accurate results (around an average of 1% better F1-scroes). This could be potentially owing to the GRU striking an architectural complexity balance between the LSTM and Naïve Bayes Classifier. Since our weekly turnip prices reach a maximum of 12 observations, the GRU does not require the additional gated memory unit of the LSTM to capture long sequence information. Instead, the GRU benefits from these shorter sequences as the issue of vanishing gradient is not exaggerated. In fact, as we have noted, for longer sequences in this time series, the conditional independency is strengthened by the fact that prior information is captured in the current observation. This further nullifies the need for having a dedicated memory cell such as in the LSTM. Additionally, as indicated by *Tables 8, A-4, A-8,* and *A-12*, the GRU boasts faster times across training, evaluation, and predictions. This is again owing to a less complex approach to approach to gated memory units, utilising only two gates (update and reset) as opposed to three gates (input, output, and forget) in the LSTM, as well as not needing a cell state to capture and transfer information, but rather utilising the hidden layer itself by completely exposing its contents.

9 CONCLUSION

This research aimed to determine the efficacy of various machine learning and sequence modelling techniques in classifying an island's turnip prices to its correct weekly trend. Furthermore, we hoped to provide this reliable trend classification as early in the week as possible. Additionally, as an extended goal, we still wished to provide some indication of forecasting capabilities.

This research has proven that sequence modelling can be a viable strategy to employ in time series classification of artificial game markets. Not only is sequence modelling a viable machine learning technique in this domain, but as empirically shown, it performs better than the probabilistic model baseline representing more conventional approaches. These gains are especially welcomed in ACNH's market, where we try to make classifications as soon as possible with limited input data, which saw desirable results achieved through the various sequence lengths. The extended goal of being able to provide some kind of forecasting ability was achieved through the more descriptive clustering analysis. This allowed the classification class itself to indicate which day and time we could expect a price spike. This project was conducted in an empirical manner, creating comparisons for two clustering techniques and three classification models, which concluded with DBSCAN and GRU being the best respective pairing for a sequence modelling-based time series classification task, whilst indicating some predictive power.

The undertaking of this project was motivated by my interest in working the *Animal Crossing* domain, which I am very fond of. The *Animal Crossing franchise* is one I have enjoyed since 2006, and sequence modelling has provided an avenue to analyse one of its mechanics (*Stalk Market*), whilst also furthering my understanding of the topic itself. I hope this body of work motivates further efforts to determine the efficacy of sequence modelling in other artificial game market domains, which has not been studied frequently.

REFERENCES

[1] R. Shrott, ‘Sequence Models by Andrew Ng — 11 Lessons Learned’, *Medium*, Jun. 09, 2021. https://towardsdatascience.com/sequence-models-by-andrew-ng-11-lessons-learned-c62fb1d3485b (accessed Jun. 09, 2022).

[2] ‘Ninji (@_Ninji) / Twitter’, *Twitter*. https://twitter.com/_Ninji (accessed Jun. 09, 2022).

[3] ‘AC:HN Turnip Price Calculator’. https://turnip-price.vercel.app/ (accessed Jun. 09, 2022).

[4] ‘The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances | SpringerLink’. https://link.springer.com/article/10.1007/s10618-016-0483-9 (accessed Jun. 09, 2022).

[5] ‘Cluster analysis - Wikipedia’. https://en.wikipedia.org/wiki/Cluster_analysis (accessed Jun. 11, 2022).

[6] K. Mysiak, ‘Explaining DBSCAN Clustering’, *Medium*, Jul. 16, 2020. https://towardsdatascience.com/explaining-dbscan-clustering-18eaf5c83b31 (accessed Jun. 10, 2022).

[7] ‘Animal Crossing: Forecasting Turnip Prices with Bayesian Inference’. https://mgold.io/2020/05/20/forecasting-turnip-prices.html (accessed Jun. 10, 2022).

[8] ‘A Tutorial on Sequential Machine Learning’, *Analytics India Magazine*, Nov. 17, 2021. https://analyticsindiamag.com/a-tutorial-on-sequential-machine-learning/ (accessed Jun. 10, 2022).

[9] J. Brownlee, ‘Gentle Introduction to the Adam Optimization Algorithm for Deep Learning’, *Machine Learning Mastery*, Jul. 02, 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/ (accessed May 11, 2022).

APPENDIX A – RESULTS FOR OTHER SEQUENCE LENGTHS

FIRST 4 PRICES

Table A - 1 - Confusion matrices for classification predictions when given first 4 turnip prices

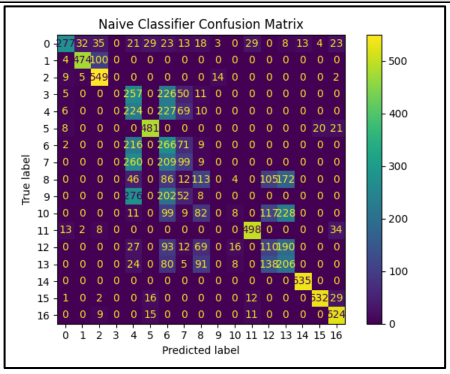
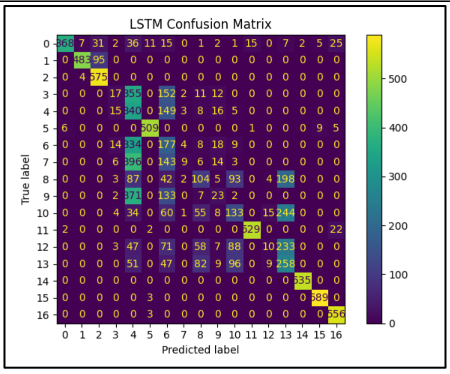
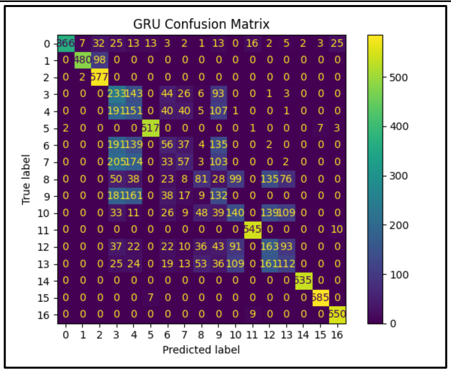
Confusion Matrix		
Naïve Bayes	LSTM	GRU
		

Table A - 2 - Summary of classification report when given first 4 turnip prices

		Weighted Average Metric		
		Precision	**Recall**	**F1-Score**
Classifier	**Naïve Bayes**	0.51	0.52	0.50
	LSTM	0.59	0.56	0.53
	GRU	0.58	0.56	0.56

Table A - 3 - Metric scores obtained when evaluating the RNN Architectures on first 4 turnip prices

		Metric					
		Accuracy	**Loss**	**MSE**	**F1-Score**	**Precision**	**Recall**
RNN Architecture	**LSTM**	0.5559	1.0607	0.0282	0.5956	0.9421	0.4405
	GRU	0.5628	1.0545	0.0280	0.5978	0.9472	0.4420

Table A - 4 - Elapsed time of each classifier during data fitting, evaluation, and prediction on first 4 turnip prices

		Stage		
		Data Fitting (s)	**Evaluation (s)**	**Prediction (s)**
Classifier	**Naïve Bayes**	0.0070	N/A	0.0029
	LSTM	234.5270	3.3299	2.6940
	GRU	214.4670	2.7120	2.6229

First 8 Prices

Table A - 5 - Confusion matrices for classification predictions when given first 8 turnip prices

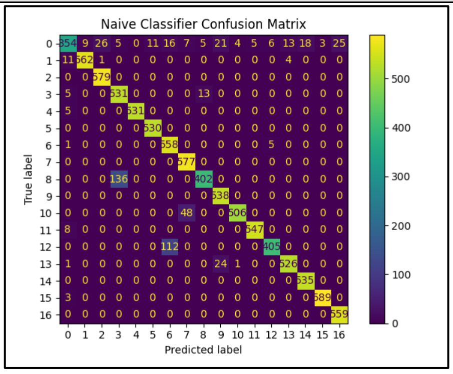
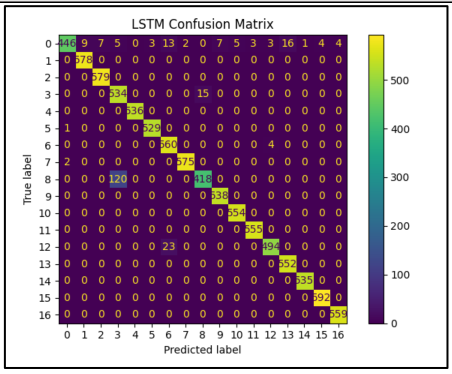
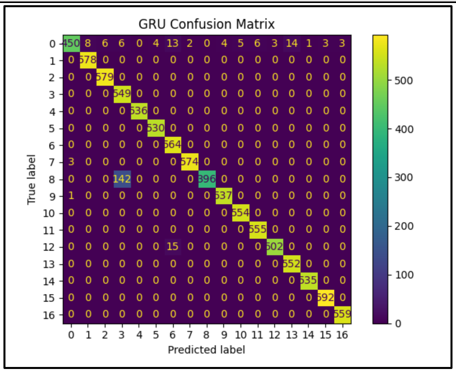
Confusion Matrix		
Naïve Bayes	LSTM	GRU
		

Table A - 6 - Summary of classification report when given first 8 turnip prices

		Weighted Average Metric		
		Precision	**Recall**	**F1-Score**
Classifier	**Naïve Bayes**	0.95	0.94	0.94
	LSTM	0.98	0.97	0.97
	GRU	0.98	0.97	0.97

Table A - 7 - Metric scores obtained when evaluating the RNN Architectures on first 8 turnip prices

		Metric					
		Accuracy	**Loss**	**MSE**	**F1-Score**	**Precision**	**Recall**
RNN Architecture	**LSTM**	0.9736	0.0813	0.0025	0.9736	0.9739	0.9733
	GRU	0.9745	0.0800	0.0024	0.9745	0.9745	0.9745

Table A - 8 - Elapsed time of each classifier during data fitting, evaluation, and prediction on first 8 turnip prices

		Stage		
		Data Fitting (s)	**Evaluation (s)**	**Prediction (s)**
Classifier	**Naïve Bayes**	0.0069	N/A	0.0030
	LSTM	240.6390	3.2559	2.5979
	GRU	221.9080	3.3100	2.5799

First 10 Prices

Table A - 9 - Confusion matrices for classification predictions when given first 10 turnip prices

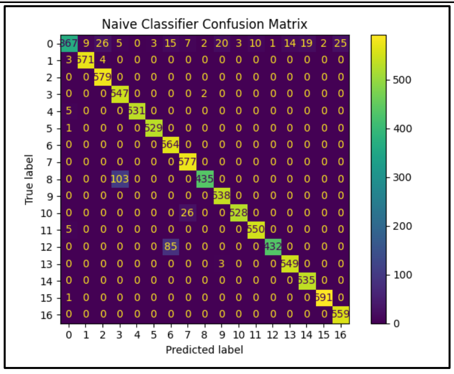
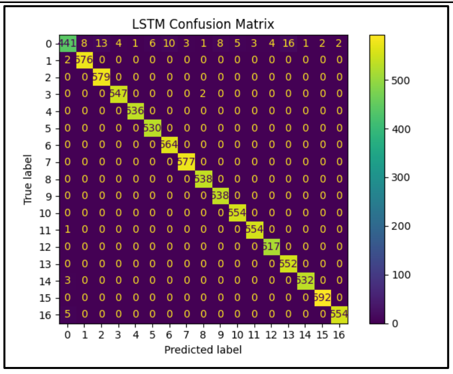
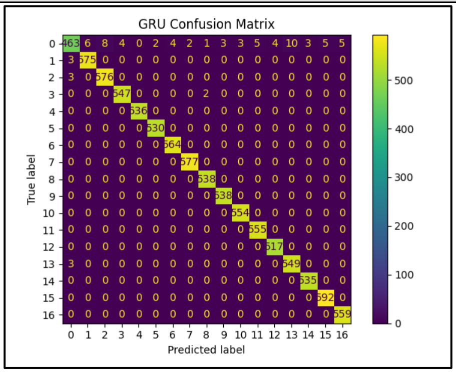
Confusion Matrix		
Naïve Bayes	LSTM	GRU
		

Table A - 10 - Summary of classification report when given first 10 turnip prices

		Weighted Average Metric		
		Precision	**Recall**	**F1-Score**
Classifier	**Naïve Bayes**	0.96	0.96	0.96
	LSTM	0.99	0.99	0.99
	GRU	0.99	0.99	0.99

Table A - 11 - Metric scores obtained when evaluating the RNN Architectures on first 10 turnip prices

		Metric					
		Accuracy	**Loss**	**MSE**	**F1-Score**	**Precision**	**Recall**
RNN Architecture	**LSTM**	0.98934	0.04101	0.0010	0.9895	0.9898	0.9892
	GRU	0.9918	0.02876	0.0008	0.9919	0.9919	0.9919

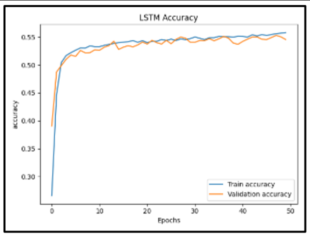
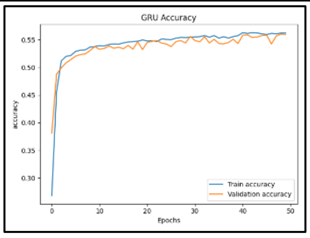
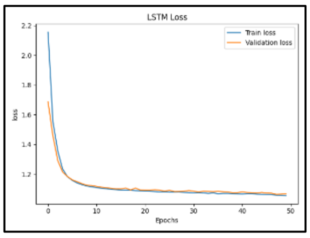
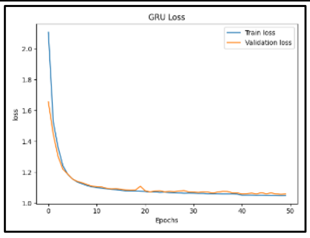
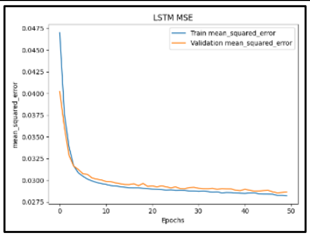
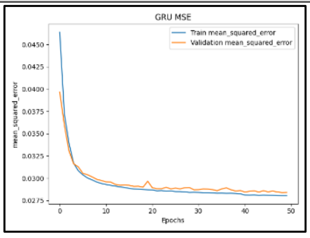
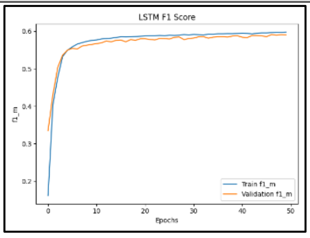
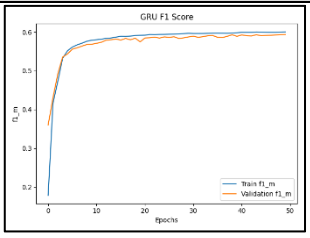
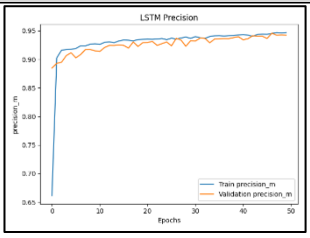
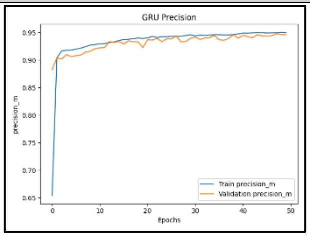
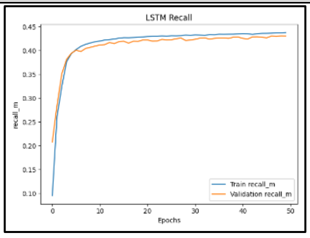
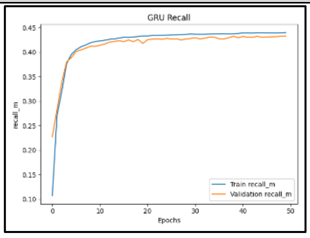
Table A - 12 - Elapsed time of each classifier during data fitting, evaluation, and prediction on first 10 turnip prices

		Stage		
		Data Fitting (s)	**Evaluation (s)**	**Prediction (s)**
Classifier	**Naïve Bayes**	0.0089	N/A	0.0040
	LSTM	253.3040	5.3789	3.5620
	GRU	230.1215	3.1939	2.6840

APPENDIX B – TRAINING AND VALIDATION METRIC PLOTS

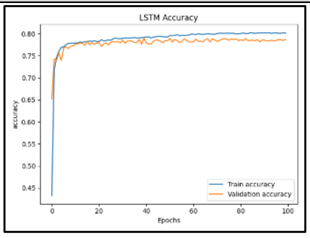
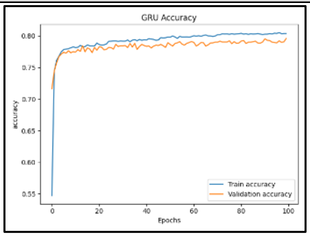
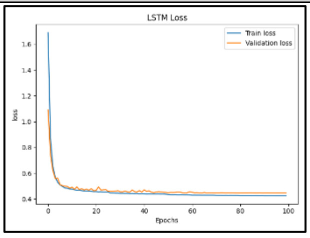
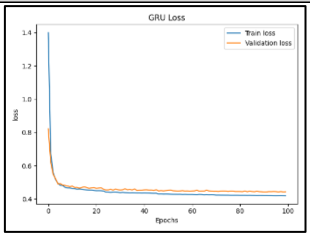
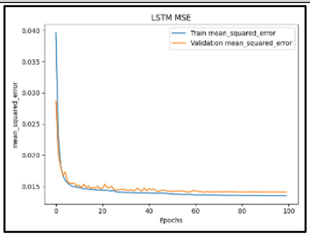
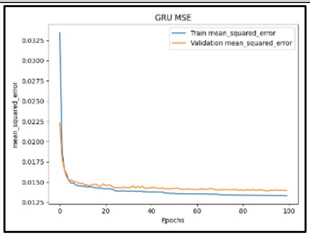
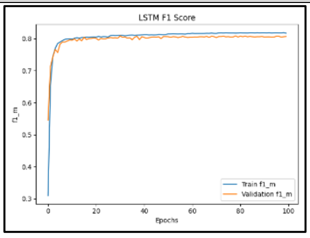
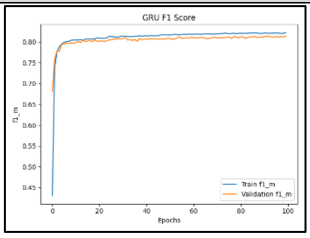
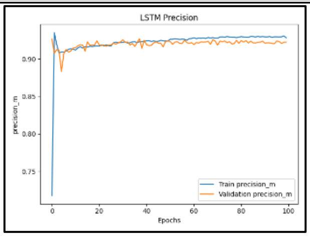
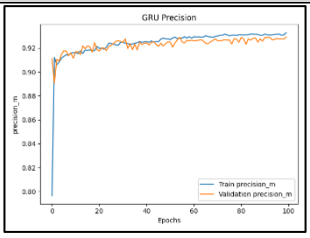
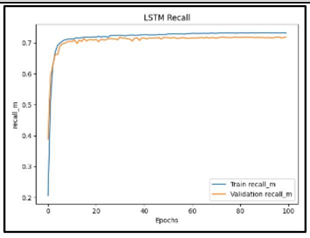
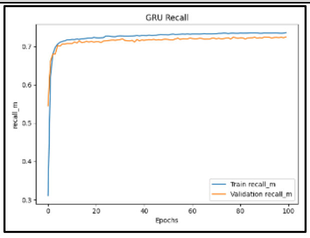
FIRST 4 PRICES

Table B - 1 - Model training and validation metric plots when given first 4 turnip prices

Accuracy		Loss	
LSTM	GRU	LSTM	GRU
			
MSE		**F1-Score**	
LSTM	GRU	LSTM	GRU
			
Precision		**Recall**	
LSTM	GRU	LSTM	GRU
			

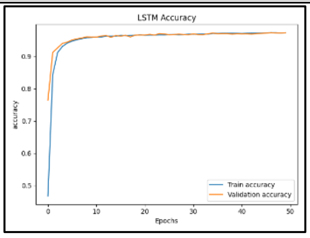
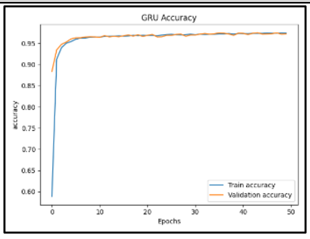
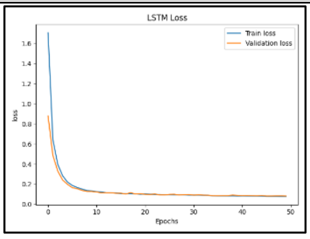
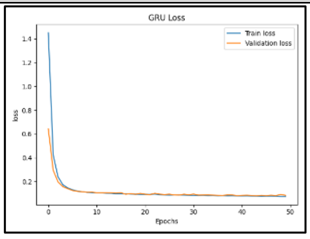
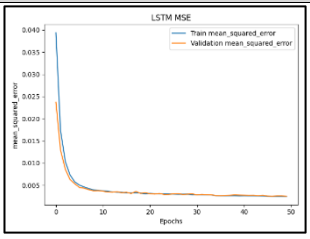
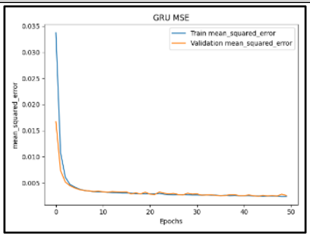
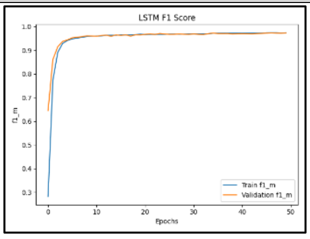
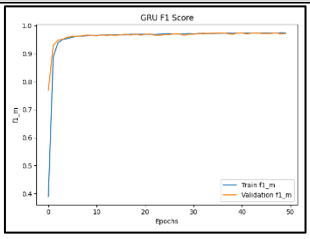
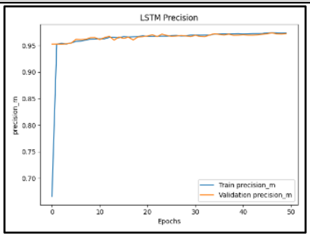
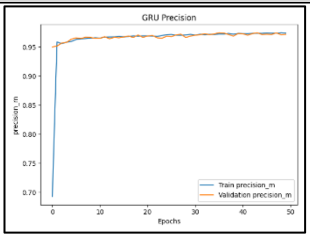
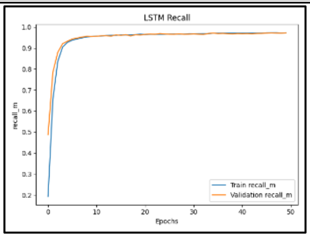
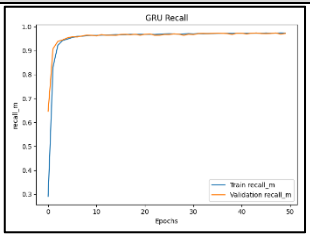
First 6 Prices

Table B - 2 - Model training and validation metric plots when given first 6 turnip prices

Accuracy		Loss	
LSTM	**GRU**	**LSTM**	**GRU**
			
MSE		**F1-Score**	
LSTM	**GRU**	**LSTM**	**GRU**
			
Precision		**Recall**	
LSTM	**GRU**	**LSTM**	**GRU**
			

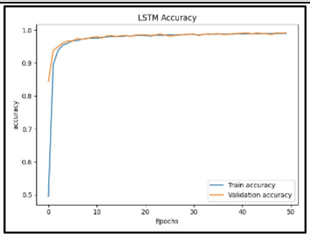
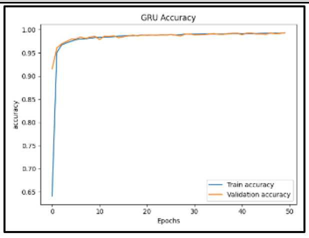
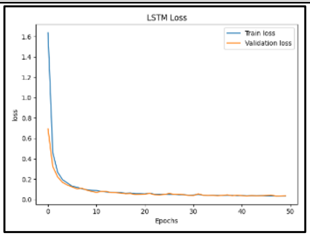
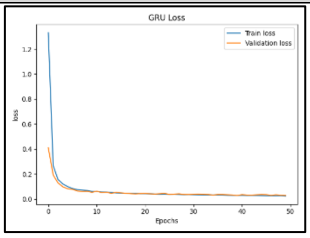
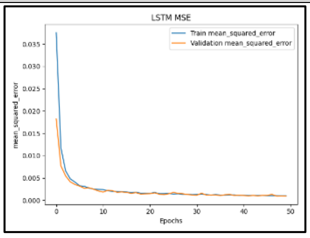
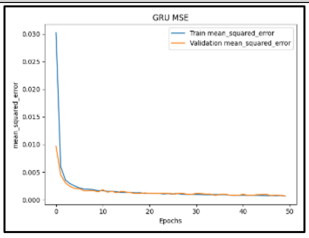
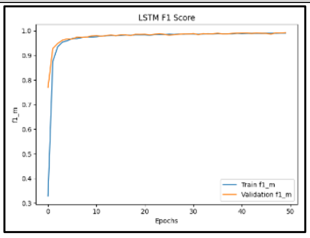
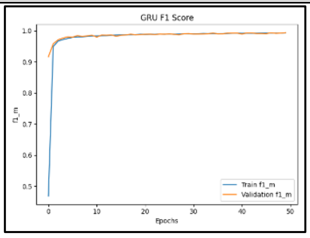
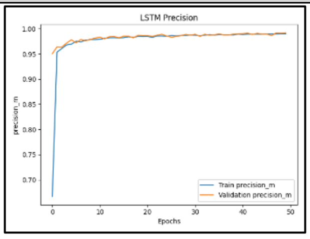
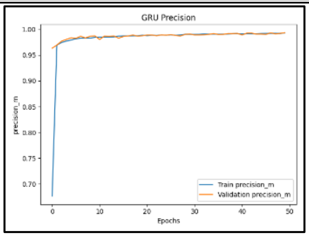
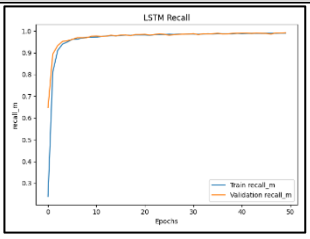
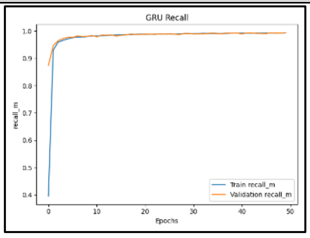
First 8 Prices

Table B - 3 - Model training and validation metric plots when given first 8 turnip prices

Accuracy		Loss	
LSTM	**GRU**	**LSTM**	**GRU**
			
MSE		**F1-Score**	
LSTM	**GRU**	**LSTM**	**GRU**
			
Precision		**Recall**	
LSTM	**GRU**	**LSTM**	**GRU**
			

First 10 Prices

Table B - 4 - Model training and validation metric plots when given first 10 turnip prices

Accuracy		Loss	
LSTM	**GRU**	**LSTM**	**GRU**
			
MSE		**F1-Score**	
LSTM	**GRU**	**LSTM**	**GRU**
			
Precision		**Recall**	
LSTM	**GRU**	**LSTM**	**GRU**
 LSTM Precision precision_m Epochs Train precision_m Validation precision_m	 GRU Precision precision_m Epochs Train precision_m Validation precision_m	 LSTM Recall recall_m Epochs Train recall_m Validation recall_m	 GRU Recall recall_m Epochs Train recall_m Validation recall_m