Classification of Financial Time Series

Samuel Lam, Dept. of Mathematics, MIT
Hoi Wai Yu, Dept. of EECS, Dept. of Mathematics, MIT

Submitted in part fulfillment of the requirements for the course
15.077 Statistical Machine Learning and Data Science

Contents

1 Introduction - Beyond our Intuition

The subject of modelling stock price is a promising one in finance. One would hope that there is an accurate model for particular stock time series, and traders can use this to make money. Often the models are not just based on the actual stock data but also our intuition on how human would behave. One very pessimistic but popular model is the random walk model, that the stock value cannot be predicted by its past values. To make formalise, if the sequence $P_0, ...P_n$ denotes the closing prices of a stock after n-days, then the random walk model says

$$X_n := \ln P_n = \mu + X_{n-1} + \epsilon_n, \quad \epsilon_n \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2) \tag{1}$$

where μ is some constant drift governed by the market, and ϵ_n is some independent, identically distributed Gaussian noise. There are so many reasons to support this heuristic. One way to illustrate is to flip many coins and record $1/2$ when getting head and $-1/2$ when getting tail. The graph of values recorded value against number of trial should have similar shape with plots of percentage daily return of typical stock [Malkiel, 1973]. [1]

We disprove the model quite easily using techniques from this class. Notice (1) can be re-arranged as followed:

$$X_n - X_{n-1} = \mu + \epsilon_n \tag{2}$$

So we expect the values of $X_n - X_{n-1}$ to follow some fixed normal distribution, i.e. the QQ-plot of the samples of $X_n - X_{n-1}$ should be a straight line. However, this was not the case when we performed this procedure on the closing price of Apple Inc. (AAPL) from 2010 to 2019 [2] In fact, the distribution is much more heavy tailed:

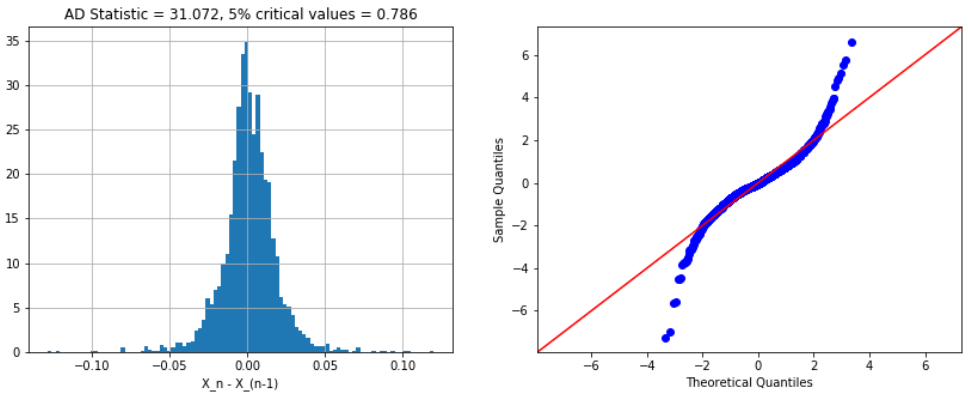


Figure 1.1: the values of $X_n - X_{n-1}$ taken from the time series of Apple Inc. stock from the beginning of 2010 to the end of 2019.

[1] Prof. Malkiel asked his student to perform the following experiment: start with a hypothetical stock of initial value $50, then determine the value of stock at next step by flipping a coin. The student doubled the closing price when head is obtained, and halved if otherwise. This is a typical experiment done during first lecture of introductory finance class when the binomial asset model is introduced.

[2] the time series comes from the CRSP US Stock and Indexes Databases, which we have used for the entire project.

The current state of the art to test the normality of a set of data is the Anderson-Darling (AD) test. [3] The unusually high value of AD test statistic confirms that $X_n - X_{n-1}$ is not normally distributed. In fact, Lo et. al. has performed a more versatile test of computing the empirical variance of $X_n - X_{n-k}, k \leq n$, which should be proportional to k if (X_n) follows (1), but unfortunately was not the case for the values of various indices and/or stock prices. [Lo et. al., 1999].

The result was a surprise to many other economists when it was first published, because it was "counter-intuitive". It revealed that the pipeline of building a model based on human intuition might be flawed, especially when the intuitions we have are inaccurate or biased. In fact, this problem propagates as we build more and more complex models based on the previous one. It is thus imperative to develop machine-driven approaches for identifying patterns inherent in stock data and through so construct a model that is less reliant on human intuitions.

Fortuitously, some of our previous research [4] involves the design of a algorithm that fits to real time series and generates synthetic ones, which can preserve short-term joint distributions up to a fixed length in the input data (Method 2 as described in Section 2). By training a classifier to classify the synthetic stock time series thus generated from real ones, we hope that the classification process will reveal some underlying structure of the time series of real stock prices. We might also hope that the simulated data can act as additional evidence, supporting or countervailing, with regards to current financial models of stock prices.

The contribution of this project is to develop procedures to distinguish the simulated data from the real data. Such procedures could then be used to evaluate the quality of any future generative models of financial time series, including the one we have developed. The criteria of distinguishing a synthetic financial time series from the real ones, including concentration of its marginal distribution and Fourier coefficients, can be used to motivate more accurate stock price models.

2 The Data

2.1 Preliminary Data Processing

For analysing the typical behaviour of stock return, to minimize confounds we decided to select stocks with high market capitalizations such that their prices are less susceptible to big changes due to illiquidity. Thus, we chose the 2500 stocks from the *CRSP US Stock and Indexes Databases* [5] that have the highest yearly-averaged market capitalizations, and which have traded for at least the full 10-year period between 2010 and 2019. Of these 2500 stocks, 2000 were chosen at random as the training set while the remaining 500 were left as the test set.

[3] Notice that we have no information about the mean and variance, so the usual Kolmogorov-Smirnov Test is not appropriate.

[4] taken as part of the Undergraduate Research Opportunities Program (UROP) under Prof. Leonid Kogan

[5] Available from `http://www.crsp.org/products/documentation/stock-data-file-layouts`; in particular, this database includes all stocks from the S&P 500 over the years

In line with existing financial theories (e.g. the random walk model), we then perform a log-transformation on the daily closing prices and take differences. We call these the **daily log-returns** of the stock. This transformation has the benefit of being timescale-invariant, such that summing the log-returns across a week is equal to the log-return computed on weekly closing prices, so that linear models are appropriate. Finally, since we wish to classify each series independently, so for each stock, we sample three 1-year segments from the time series at random starting days during the full 10-year period, so that there is no unwanted confound of time-alignment between the series; we will call these 1-year (252-day) series the **stock return series**.

2.2 Synthetic Data Generation

In order to train our classification algorithms to identify properties of real stock return series, we must provide counterexamples for the supervised learning task. However, these counterexamples must be well-designed – if they are not designed to be superficially similar to the real stock series, the algorithms are likely to learn trivial properties. For example, if our synthetic data has too small a scale compared to the real stock return series, the classification algorithms will find the uninteresting property that real stock returns have larger magnitudes on average.

We thus propose to analyse the performance our classification algorithms when the real dataset is compared against each of the following two sets of synthetic data:

- **Method 1:** Generate synthetic data by sampling random log-return values i.i.d. from a fixed randomly selected stock, and joining 252 of these randomly sampled values into a time series. This random stock was selected within the training set for training the classifier, and within the test set for testing the classifier.

  This acts as our the baseline comparison – the method guarantees that the synthetic time series has the same empirical cdfs as a real series in expectation, while losing all information about trends and correlations in the real stock series. Thus, the classification algorithm classifying this synthetic data from real data should reveal patterns about temporal trends in stock returns.

- **Method 2:** This is an efficient model resulting from our earlier research in a UROP. The synthetic data $X'_1, \dots, X'_{252}$ is generated from a hidden Markov model fitted to a randomly selected stock, where the underlying process is an n-gram model with the n-dimensional joint distributions estimated by bootstrapping the empirical joint distribution and smoothing it with a data-dependent Gaussian kernel:

$$\begin{aligned} X'_1, \dots, X'_n &\sim \mathbb{P}_{X_1,\dots,X_n} \\ X'_{m+n} &\sim \mathbb{P}_{X_n|X_1,\dots,X_{n-1}}(\cdot \mid X'_{m+1}, \dots, X'_{m+(n-1)}) \quad (m \geq 1) \\ \mathbb{P}_{X_1,X_2,\dots,X_n} &= \text{smoothed}(\hat{\mathbb{P}}_{X_1,X_2,\dots,X_n}) \end{aligned}$$

  We will not go into the details of this method, but mathematical analysis shows that the generated data is stationary and we can show that the joint distributions of the generated data closely matches the original data on intervals of length up to n trading days. Following our UROP research, we choose $n = 10$ (i.e. two weeks) as the best model to compare here.

We know that this model should preserve more information about short-term trends of the stock log-returns compared with Method 1, while retaining the property that the marginal distributions are identical. Thus we expect classification performance to be much worse on this dataset as it is much closer to a real dataset. By comparing differences in classification accuracy between using each method to generate counterexamples, we can hopefully identify the classification methods that identify short-term trends, and if any classification methods can well distinguish Method 2 data from real data, we expect this classifier to capture information about longer-term trends in the data.

Below, we will use **Task 1** and **Task 2** as shorthand to directly refer respectively to the task of classifying real data from Method-1 synthetic data, and the task of classifying real data from Method-2 synthetic data. Since we have $2000 \times 3 = 6000$ real time series, we generate a corresponding 6000 synthetic time series with each method, for a total of 12000. The dataset for each Task thus has shape $(12000, 252)$.

3 Discrete Fourier Transform

From now on, we define Y be the label of the time series, which is 1 if the time series is from real data and 0 otherwise.

3.1 Feature Generation

Since we assume our data is stationary, the specific position of the stock log-returns in the time series is not meaningful outside of the relative position (which matters for the autocorrelation structure). Moreover, the data can be reasonably expected to have periodic factors corresponding to factors like seasonality. Thus, it makes sense to convert our data from the time domain to the frequency domain.

The **Discrete Fourier Transform** (DFT) is a natural technique for extracting frequency features from a time series. For the 252-day series ($N = 252$), the transform is defined by

$$F_k = \sum_{n=1}^{N} \exp\left(-2i\pi \frac{kn}{N}\right) X_n,$$

where F_k is the coefficient of the component corresponding to period k. These coefficients are evidently complex-valued. To analyse these with standard classification algorithms, we can consider real **proxies** $\tilde{F}_k$ such as (1) the real part, (2) the imaginary part, (3) the absolute value/modulus, or (4) the angle/argument of the complex numbers, or (5) all four of these real values concatenated together.

3.2 Linear Models

We conduct a (ℓ^2-regularised) logistic regression on various choice of the proxies, assuming the following logistic model

$$\ln\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \sum_{i=1}^{k} \beta_k \tilde{F}_k + C\epsilon, \tag{3}$$

where π is the probability that the true label Y is 1, i.e. the probability that the time series comes from the log-returns of a real stock, and ϵ is a Gaussian noise term.

We conduct 5-fold cross-validation for the regularization constant C. Table 3.1 summarises the train and test accuracies for the best value of C and different choices of real proxies $\hat{F}_k$, as discussed in Subsection 3.1.

	Task 1		Task 2	
Proxy ($\tilde{F}_k$)	Training	Testing	Training	Testing
Real	0.53	0.49	0.54	0.49
Imaginary	0.54	0.50	0.53	0.52
Modulus	0.69	0.70	0.65	0.65
Argument	0.55	0.49	0.54	0.50
All	0.71	0.68	0.66	0.61

Table 3.1: Training and testing accuracy for various proxies when classifying each of two sets of synthetic time series from the real time series using logistic regression.

Note that the real part, the imaginary part, and the argument of the coefficients all provide no predictive power with testing accuracy around 0.5. Only when the modulus is used in the model is the classification accuracy non-trivial; and we note in addition that using all 4 proxies in fact decreases the testing accuracy, indicating that the other 3 proxies do not help at all. This indicates that using attributes other than moduli is not useful. Since the modulus of a Fourier coefficient indicates the magnitude of the corresponding periodic component, we conclude that the **magnitude** of the periodic trends is the most important extracted feature.

The respective values of the regularization constants using the modulus proxy for each Task is 0.316 and 0.794. The sensitivity and specificity of the models can be found in Table 3.3.

Note also that the train/test scores for Task 1 is are higher than that for Task 2 by around 5%. This indicates that Method-1 synthetic data is slightly easier to classify using the DFT linear model than the Method-2 synthetic data, which suggests that Method-2 data preserves more of the periodic patterns and trends in the real dataset, which is known by the fact that patterns on time intervals of up to 10 days must be preserved as mentioned in Section 2. However, since the accuracy is still very high for Task 2, this indicates that Method 2 is not successful at capturing most of the longer-term trends. Further checks are required.

Principal Component (Logistic) Regression (PCR) Running a full regression here is very time consuming, and moreover not very informative as the Fourier coefficients are quite homogenous. However, for a typical dataset, the Fourier coefficients are clustered because the existence of any periodic trends will lead to a bump in many close coefficients. We thus hope to identify trends and reduce the dimensions of the data by conducting PCA. For our usecase of PCR, we estimate the principal components

of data by singular value decomposition, project onto those principal components and perform logistic regression. We conduct this methodology with varying number of principal components and to see how many principal components are sufficient.

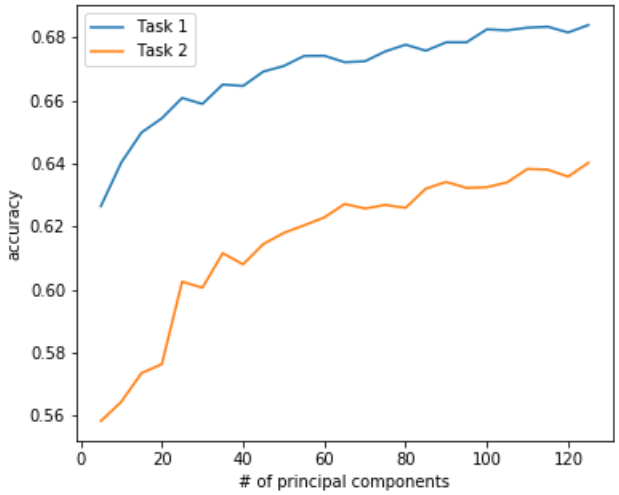


Figure 3.1: Testing accuracy vs. number of components used in PCR for the two Tasks

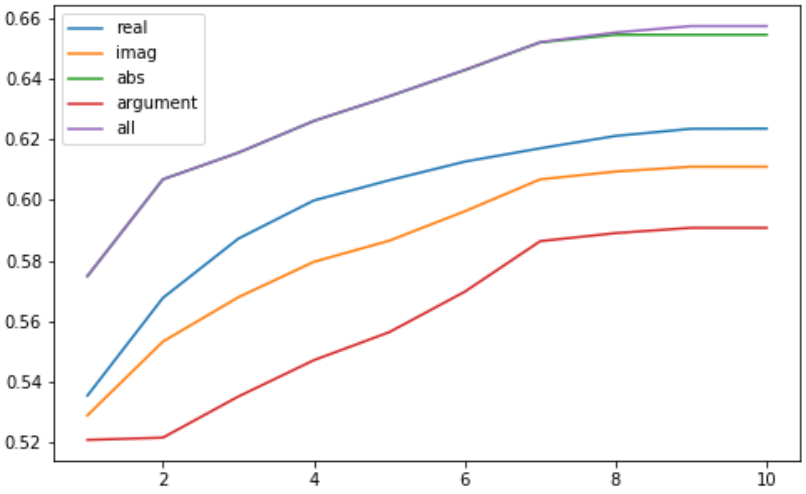
Unfortunately, there is not a clear ”elbow” in the figure. Moreover, the maximum test accuracies (0.68 for Task 1 and 0.64 for Task 2) never exceed that of full logistic regression. This signifies that even later components have significant contributions to the model and PCR is **not effective** in reducing the dimensionality of the data.

3.3 Tree-based Classification

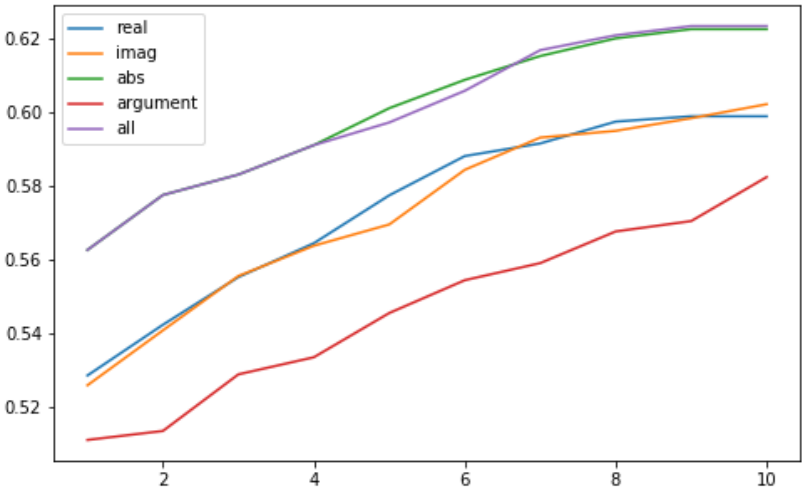
Another method of classification on the DFT features are tree-based classifiers. We hope to capture non-linear patterns, in contrast to the above linear models, using this method. While the DFT features are in of themselves not very informative, this also serves as a convenient contrast for the next set of features we consider in Section 4.

We analyse the testing accuracies of decision tree classifiers of various depths, applied on various proxies of the Fourier coefficient, in Figure 3.2 The best depth for the classifier for each Task appears to be depth 7. We find that while the real and imaginary parts have some non-linear relationship with the real/fake classification, in the end the modulus is most consistent predictor, reaffirming that the size of the periodic trends matters.

We note that the accuracies of the best decision tree classifiers remains low. To increase accuracy we attempt random forest classifiers of same depth as the previous tree (in our case depth 7). While the accuracies improve, the accuracies are still generally below that of the linear logistic regression models. It is also apparent that random forest is suffering from overfitting, by looking at the testing accuracies shown in Table 3.2. This indicates that perhaps a combination of many more features is needed for good classification, in accordance with the observation from PCR above.



(a) Task 1



(b) Task 2

Figure 3.2: The two figures plot the testing accuracies (y-axis) against the depth of trees.

	Task 1		Task 2	
Proxy ($\tilde{F}_k$)	Training	Testing	Training	Testing
Real	0.83	0.64	0.86	0.61
Imaginary	0.83	0.63	0.87	0.59
Modulus	0.86	0.67	0.92	0.62
Argument	0.98	0.51	0.99	0.51
All	0.87	0.67	0.92	0.62

Table 3.2: Training and testing accuracy for various proxies when classifying each of two sets of synthetic time series from the real time series using a random forest.

We have previous discussed the accuracies of the models; we also summarize the sensitivity and specificities of the models in this section in Table 3.3. We find that as mentioned, the logistic regression method has the best performance overall in each area of comparison for both tasks.

	Task 1		Task 2	
Method	Sensitivity	Specificity	Sensitivity	Specificity
Logistic regression	0.67	0.69	0.71	0.59
Decision tree	0.60	0.63	0.58	0.56
Random forest	0.62	0.71	0.56	0.65

Table 3.3: Sensitivity and specificity of the trees and linear models used.

4 TSFresh

4.1 Feature Generation

Since the above methods overall do not have particularly good performance, we decide to utilize **tsfresh**, a Python library that extracts features of time series data useful for cross-domain analysis. This is suitable for our needs since we want to extract features in a manner independent of human intuition in the economics or finance fields for machine analysis.

We use the default settings of the tsfresh package, which generates 781 features [6] on the 252-dimensional data. Notable features include various quantiles, coefficients of the DFT, coefficient of ARIMA, linear trends, and various aggregate metrics such as the average of the maximums computed on short windows. We note that since DFT coefficients are contained within the components of these features, the models generated using these features are expected to have strictly stronger predictive power than in Section 3. Moreover, we note that these features are overall more human-interpretable than the DFT features, which suggests using sparse methods to interpret the resulting model.

[6] Available in `https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html`

4.2 Linear Models

We start with ℓ^2 regularized logistic regression, using 5-fold cross validation for the value of the regularization constant, as discussed above. Table 4.1 summarizes the training and testing performance for the best value of C for each Task.

Metric	Task 1	Task 2
Training accuracy	0.91	0.77
Testing accuracy	0.88	0.74
Regularization constant C	10	1e4
Sensitivity	0.87	0.75
Specificity	0.87	0.75

Table 4.1: Results of the logistic regression with cross-validated regularization constant for each classification task.

To subsample features and create a more human-interpretable model, we attempted ℓ^1 regularized logistic regression, using 5-fold cross validation for the value of the regularization constant. However, logistic regression with ℓ^1 regularization proved extremely slow using Python's sklearn module on our large 12000×781 dataset, and we could not get the algorithms to reach convergence. To compensate, we attempted Lasso regression, which is similar to ℓ^1 regularized logistic regression except we use squared distance from the 0-1 classification label instead of a cross-entropy loss as the loss function, again with 5-fold cross-validation for the regularization constant. The results are summarized in Table 4.2.

Metric	Task 1	Task 2
Training accuracy	0.88	0.75
Testing accuracy	0.87	0.75
Regularization constant C	1e-3	1e-3
Number of nonzero components	460	496
Sensitivity	0.86	0.77
Specificity	0.88	0.73

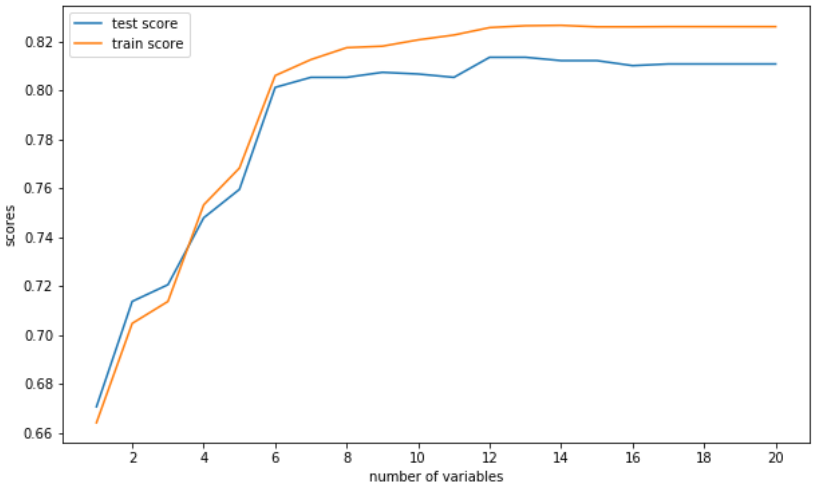
Table 4.2: Results of the Lasso regression with cross-validated regularization constant for each classification task using the 0-1 classification labels as Y.

We note that the test performance is comparable to the ℓ^2 regularized logistic regression, but the set of nonzero components is extremely large (recall that the tsfresh feature set contains only 781 features). As a result, this model is not human-interpretable. To rectify this issue, we decided to conduct forward selection on features to select a small feature set.

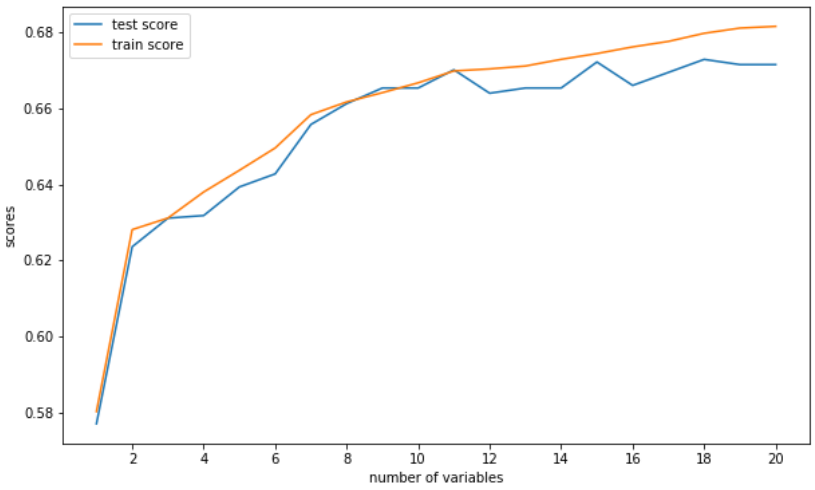
Forward Selection Starting with the empty set as the selected feature set, we determine the accuracy of the model resulting from adding each other feature individually to the feature set by 5-fold cross validation, and after all features have been considered, we select the feature with the best validation accuracy during cross-validation. We repeated this until we have a set of 20 features. We then determine the accuracy of a sparse ℓ^2-regularized logistic regression model using only the first n features, for n

from 1 to 20, again using 5-fold cross-validation for the regularization constant. The results are as shown in Figure 4.1.

We see an "elbow" in the testing accuracy for Task 1 around 6 features, and an elbow near 9 features for Task 2, beyond which the model mostly seems to overfit with the training accuracy drifting away from the testing accuracy. The indices of selected ts-fresh features (note that the order of the features is deterministic), and the names of the top features, are reported in Table 4.4.



(a) Task 1 (Elbow = 6)



(b) Task 2 (Elbow = 9)

Figure 4.1: The two figures plot the testing accuracies (y-axis) against the number of features selected when downsampling the data.

We find that the testing accuracy of logistic regression when using this downsampled small set of features is significantly worse than when including all features; however, the feature sets are small enough to be human-interpreted.

Metric	Task 1	Task 2
Training accuracy	0.81	0.66
Testing accuracy	0.80	0.67
Number of nonzero components	6	9

Table 4.3: Training and testing accuracy for the two tree-based classifiers for each task.

Task	Indices	Name in tsfresh
1	1	0__ratio_beyond_r_sigma__r_1
	2	0__augmented_dickey_fuller__attr_"usedlag"__autolag_"AIC"
	3	0__change_quantiles__f_agg_"mean"__isabs_True__qh_0.6__ql_0.0
2	1	0__lempel_ziv_complexity__bins_100
	2	0__matrix_profile__feature_"75"__threshold_0.98
	3	0__change_quantiles__f_agg_"var"__isabs_True__qh_0.6__ql_0.4

Table 4.4: The top 3 features for each classification task in the forward selection method. For Task 1, the selected 6 features are numbers 748, 733, 213, 233, 249, and 682; for Task 2, the selected 9 features are numbers 747, 763, 780, 242, 213, 87, 749, 77 and 229.

For Task 1, the most informative feature is the ratio of datapoints in the time series beyond 1 standard deviation around the mean, which is a measure of tailedness of the sampled time series; since the real time series are more trendy, it makes sense that the outliers for those time series are more clustered temporally; on the other hand, the Method 1 of synthetic data generation simply samples points from the whole history of the stock with no consideration of temporal trends. Thus we do expect that the real data in this case to be more likely to have heavier tails than the synthetic data, as indicated. We note that this is likely to happen to Method 2 data too, but to a lesser extent as the data is clustered in the short-term similarly to the real data.

For Task 2, the most informative feature is the Lempel-Ziv complexity, which is a measure of the repetitiveness of substrings of the time series. Since the Method 2 of synthetic data generation uses a hidden Markov model where the n-gram transition kernel is based on the bootstrap estimate of the true joint distribution of the data, it makes sense that the simulated time series would repeat certain sequences of length n or less in parts of the distribution that were encountered multiple times in the simulation process but are rare in the real data. In fact, during our UROP, we independently discovered a similar problem (low entropy of the sampling process, a part of data generation) that we have already attempted to address, but this is evidence that the problem has not been completely rectified.

Given more time, we can analyse the effect of the remaining important predictors as well. This brief analysis serves at least as indication that the tsfresh features combined with forward selection of features is informative as to the issues with each model.

4.3 Tree-based Classification

To capture non-linear patterns, we once again use tree-based classifiers. We analyse the testing accuracies of decision tree classifiers of various depths, as shown in Figure 4.2. The best depth for the classifier for each Task appears to be depth 8 and depth 10 respectively. We also can inspect the first few nodes of each classification tree, as shown in Figure 4.3.

For Task 1, the top node asks whether the ratio of datapoints in the time series beyond 1 standard deviation of the mean is less than 0.454; we find that this is true of most of the synthetic data and false of most of the real data, in accordance to our previous analysis. This is a highly effective classifier; after 1 node the classification accuracy is already up to 0.67. For Task 2, the top node asks whether the ratio of datapoints in the time series beyond 0.5 standard deviations of the mean is less than 0.561; we find that this is true of most of both types of data, but more likely to be false of the real data than the artificial data, and the classification accuracy after this node is much lower at 0.59. Again this is in accordance with our previous analysis of the clustering effect being alleviated somewhat with the preservation of local correlations, but trends over the whole 252-day period are still unaccounted for.

Once again, to increase accuracy we attempt random forest classifiers of same depth as the previous trees. While the accuracies improve, once again the accuracies are still generally below that of the linear logistic regression models, and the method suffers from overfitting, by looking at the testing accuracies shown in Table 4.5.

	Task 1		Task 2	
Method	Training	Testing	Training	Testing
Decision tree	0.79	0.78	0.68	0.63
Random forest	0.86	0.81	0.93	0.69

Table 4.5: Training and testing accuracy for the two tree-based classifiers for each task.

We once again summarize the sensitivity and specificities of the models in this section in Table 4.6. We see that the Lasso and the logistic regression methods are comparable as the best methods in each metric and each task. However, the forward selection and the tree methods both achieve sparsity and human interpretability where the previous two methods do not. Of these methods, the forward selection algorithm appears the best, being able to downsample to very few features while keeping most of the performance, and still leading to interesting interpretable findings as we have discussed above.

5 Conclusion

We have therefore surveyed a few possible ways in distinguishing the synthetic time series from the real time series. In fact, all algorithms we have used so far agree with the fact that the data generated from Method 2 is harder to be distinguished from the real data than those generated from Method 1; but we have identified systematic

	Task 1		Task 2	
Method	Sensitivity	Specificity	Sensitivity	Specificity
Logistic regression	0.87	0.87	0.75	0.75
Lasso	0.86	0.88	0.77	0.73
Forward selection	0.82	0.79	0.70	0.63
Decision tree	0.75	0.79	0.64	0.62
Random forest	0.77	0.85	0.73	0.65

Table 4.6: Sensitivity and specificity of the trees and linear models used.

discrepancies between real stock data and our synthetic data in each case using interpretable methods on the informative tsfresh features. We notice that the use of a much larger feature set would result in a significant improvement of classification performance (by around 10% for both sensitivity and specificity). However, the use of such feature inevitably impose our intuition that the tsfresh features are fundamental building blocks of the data generation (hence also economic theories based on our data), which might be arguably against our initial goal of discovering human-bias-free models. Nevertheless, this could be an important stepping stone in formulating better classification methods, or conversely better models of stock data and thus better data generation methods. We will use the procedures here to evaluate the quality of other data generation methods developed in our research.

As a final note, we note that there are some other possible ways to improve the algorithms. One main direction is to try incorporate neural network architectures. The usage of convolutional neural network on spectrogram to classify time series had been well developed. However, the method worked the best only for non-stationary series, which is not the case for the project. It is therefore not clear how effective will the approach be. One further ambitious idea is to train a generative adversial network (GAN), which allows the computer to learn how to generate and classify data at the same time in a competitive manner. However, the difficulty of training a GAN is well beyond our ability.

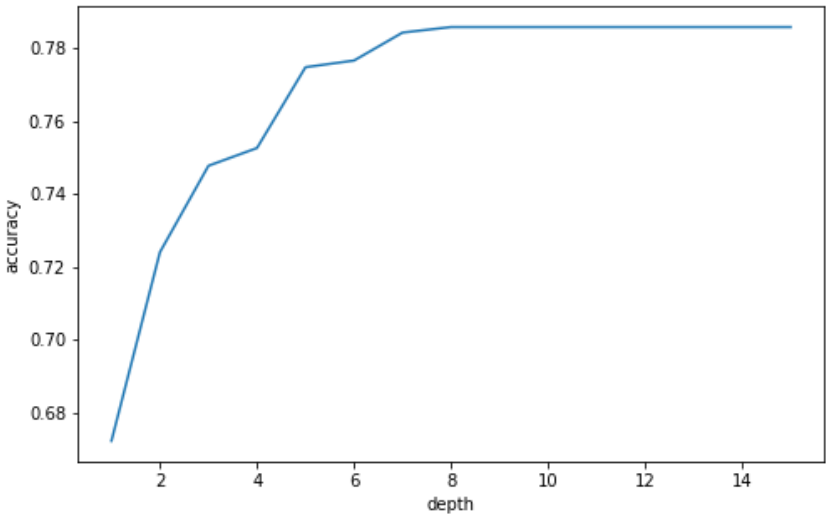
References

[1] A. W. Lo and A. C. MacKinlay, *A Non-Random Walk Down Wall Street*. Princeton University Press, 2011.

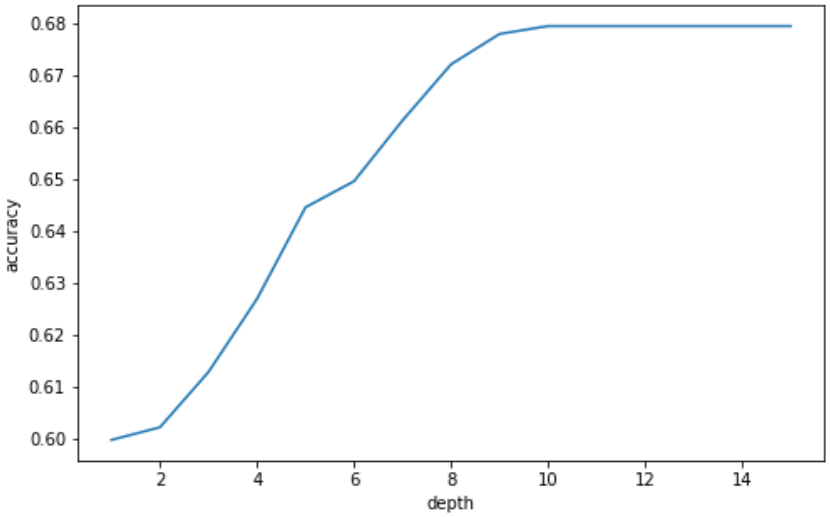
[2] B. G. Malkiel, *A Random Walk Down Wall Street*. W.W. Norton amp; Company, 2003.

[3] T. Hastie, J. Friedman, and R. Tisbshirani, *The Elements of statistical learning: data mining, inference, and prediction*. Springer, 2017.

[4] “Speech command recognition using deep learning.” [Online]. Available: https://uk.mathworks.com/help/deeplearning/ug/deep-learning-speech-recognition.html

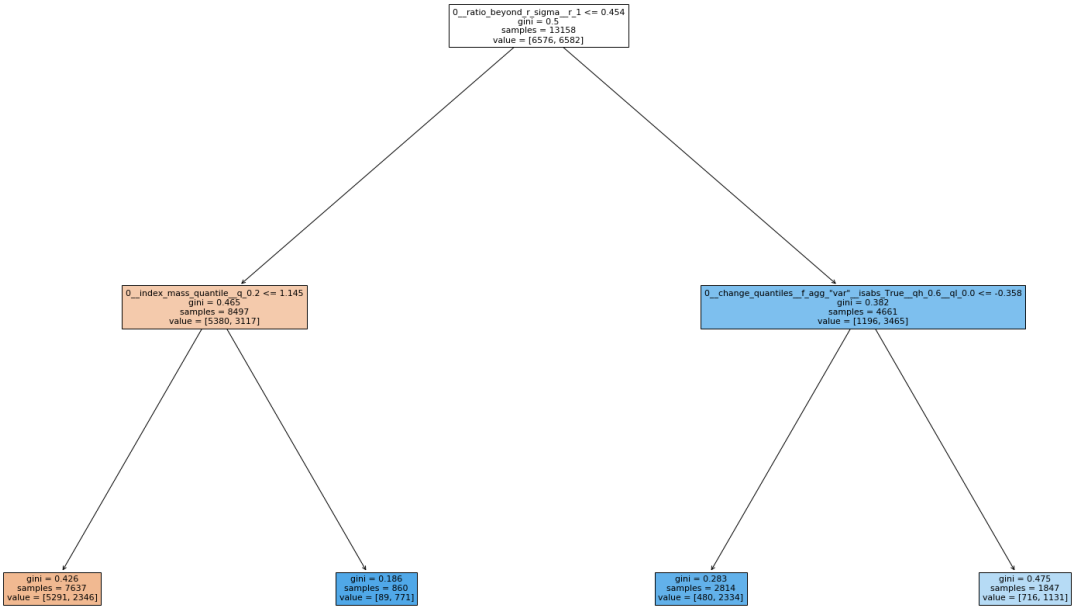


(a) Task 1 (Elbow = 8)

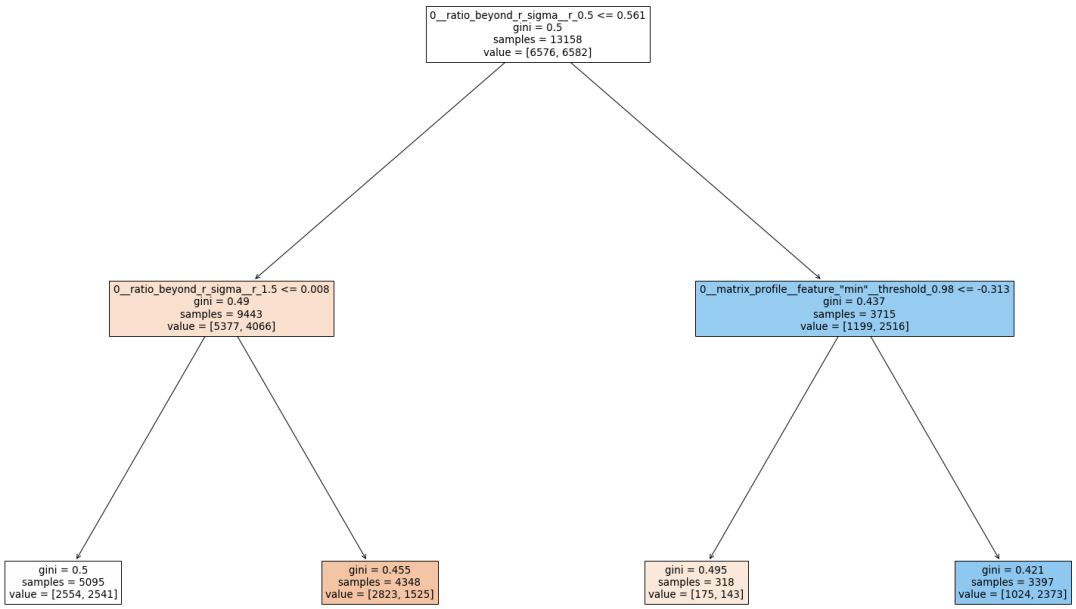


(b) Task 2 (Elbow = 10)

Figure 4.2: The two figures plot the testing accuracies (y-axis) against the depth of trees for tsfresh data.



(a) Task 1



(b) Task 2

Figure 4.3: The first two layers of the decision trees for each task using tsfresh features.