Comparative study of conformal and bootstrapping predictors evaluating predictive uncertainty

Anonymous

Submitted for the Degree of Master of Science in

Your MSc Programme

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 10,695

Student Name: Anonymous

Date of Submission: 5/9/2025

Abstract

Uncertainty quantification is essential for building trust in machine learning models especially when assigned to a critical system whose failure can lead to loss of environmental and humanitarian life, loss of assets, and security breaches. This study presents a comparative evaluation of two prominent uncertainty quantification estimation techniques: bootstrapping and conformal predictors (MBB and Mondrian CP) using a consistent model Mqf^2 and a curated Nvidia stock dataset framework of a time series setting. I assess each predictor's ability to produce valid and efficient prediction intervals by examining marginal coverage, conditional coverage (feature stratified coverage and size stratified coverage) and region size. My analysis show that MCP tends to produce overall better metrics for both validation and prediction efficiency when the confidence level alpha is 0.2, but MBB provides better prediction intervals when alpha is 0.1 and applied to an indicator dataset for the Nvidia stock.

Contents

1 Materials and metrics

1.1 Introduction

Accurate quantification of predictive uncertainty is a desirable measurement especially for critical systems. Critical systems deliver choices that lead to consequences on human life, major financial decisions, and mission critical where goals rely on high reliability to ensure failure is mitigated and held accountable if made.

Conformal prediction and Bootstrapping provide a framework that can be applied to any model with minimal assumptions making both versatile options when calculating model metrics. This versatility allows it to be applied to both Classification and Regression problems, parametric models with underlying distributions and importantly non-parametric models freely emphasising more complex relationships allowing a wider variety of domains.

Metrics directly assess the model accuracy through coverage uncertainty measures and interval validation producing well calibrated trustworthy results with the ability to show bias and outliers informing the users of the underlying model, areas of unreliability.

1.2 Conformal prediction

1.2.1 Introduction

Conformal prediction builds upon an existing prediction model, manipulating the model to produce predicted values which it uses to produce conformity/ non-conformity scores. It then calibrates the model based on the user adjustable confidence level to return prediction intervals to calculate metrics including rigorous marginal coverage guarantees through the exchangeability assumption. Under different contexts and applications, many methodological variations tailor to specific situations mainly trading prediction efficiency with computational efficiency. I will explain 3 conformal prediction branches: Transductive Conformal prediction, Split Conformal prediction and Cross Conformal prediction with the marginal coverage and exchangeability assumption.

1.2.2 Transductive conformal prediction

The first implementation of conformal predictors is the Transductive conformal prediction method and aims to maximise the prediction efficiency making the prediction intervals as small as possible however is known to be expensive in computational efficiency.

Given a training set with a set of features x_i and a respective label y_i and a test set consisting of features x_j for which the labels must be predicted and given a prediction interval to, this can be formulated as $z_1 = (x_1, y_1), \dots , z_l = (x_l, y_l)$ and $x_{l+1}, \dots , x_k$ respectively. The key to Transductive conformal predictors is their transductive non-conformity scores which can be noted as a function mapping the training set and each object in the test set to a real number describing the lack of conformity between the test set and the training set A: $Z^* \times Z^* \rightarrow R$. Given the set of unique labels from the training set $(v_1, \ldots, v_k) \in Y^k$, we assign the test object each label $y_j := v_j - l$ and $z_j := (x_j, y_j)$ for $j = l+1, \ldots, l+k$ and compute its transductive non-conformity score. Essentially, given a classification example where the training set contains labels of only cat, dog, and mouse, we make test objects of (x_{l+1}, cat), (x_{l+1}, dog) and (x_{l+1}, mouse) and compute the non-conformity score to show how related x_{l+1} is to each of those labels comparative to the training set hence $\alpha_S := A(z_{\{1,\ldots,l+k\}\setminus S}, z_S)$ produces the set of non-conformity scores for the entire scenario above. From the set of non-conformity scores, we can produce the p-values for the test object per y label:

$$p(v_1, \ldots, v_k) := \frac{|\{S \,/\, \alpha_S \geq \alpha_{(l+1,\ldots,l+k)}\}|}{(l+k)!/k!},$$

p-value calculation [1]

We repeat this, generating p-values for every subsequent test object in the test set individually. This is the main reason this process is extremely computationally intensive since we must calculate individual p-values for each label in training's Label set multiplied by the number of test objects in the test set hence it doesn't scale very well with increasing test size and especially increasing label set size for the case of classification.

Using the p-values and a user specified confidence level $1 - \epsilon$, we can extract the prediction set for each test object and label pair taking the label of the training set label object of high conforming p-values larger than the 1 – confidence level.

$$\Gamma^{\epsilon}(z_1, \ldots, z_l, x_{l+1}, \ldots, x_{l+k}) := \{(v_1, \ldots, v_k) \in Y^k \,/\, p(v_1, \ldots, v_k) > \epsilon\}$$

Prediction region calculation [1]

If the transductive conformal predictor is built upon an existing model, it will be incorporated with the non-conformity score implemented by finding the difference between the results of the estimates with each label in the Label set adding another level of computational inefficiency since the model will need to be retrained multiple times.

Given the prediction set under an uncertainty quantification, validity and efficiency metrics can be calculated for further analysis.

1.2.3 Split conformal prediction

Split conformal prediction is widely used and extremely popular having even further variants due to its comparatively high computational efficiency to Transductive conformal prediction. Adapting Transductive conformal prediction with an adjustment in the splits of the dataset for calculating non-conformity scores means it reduces the number of times the model is retrained.

It splits the dataset $(X_i, Y_i)_{i=1}^{n}$ into 3 sets: train, calibration and test, training an underlying model μ^ using (X_{train}, Y_{train}) to provide, in the case of regression, prediction points used for calculating non-conformity score residuals $\hat{s}_{train}(x, y) = |y - \hat{\mu}(x)|$ using the calibration set (X_{cal}, Y_{cal}). Hence, we can compute the $1 - \alpha$ empirical quantile for these residuals where α is the confidence level. For each i in X_{test} we can compute the prediction intervals

$$C_{1-\alpha}(X_i) := \{y \in Y : \hat{s}_{train}(X_i, y) \leq \hat{q}_{1-\alpha}\}.$$

Prediction interval calculation [10]

The prediction intervals will still require each test object to be combined with every label in the training and calibration Label set like Transductive conformal prediction however the residuals are now compared with the empirical quantile of the confidence level rather than calculating p-values to compare to the significance level.

From the prediction intervals marginal, size-stratified, feature-stratified, other conditional coverages can be calculated to test the validity of the model alongside average interval width and region size can be calculated to test efficiency. These simple steps make split conformal prediction a popular option with high computational efficiency by only training the model once to extract residuals, especially suitable for scalability besting other techniques even outside conformal prediction i.e. Transductive CP and Bootstrapping routines. However, the predictive intervals tend to be wider since the calibration set is a split from the training set using less values to calculate the non-conformity scores with the test values meaning the empirical quantile results in wider intervals reducing efficiency to maintain marginal coverage guarantees.

1.2.4 Cross Conformal Predictors

Split conformal predictors aim to be more computationally efficient than most other predictors including Cross conformal however, trades this with prediction efficiency since it only uses the training set to generate prediction rules and the calibration set for non-conformity scores rather than using the whole training and calibration set for both jobs as done by Transductive conformal prediction. Cross conformal builds on Split conformal predictors and Transductive conformal prediction using folds to use both training and calibration sets for prediction rules and non-conformity scores without retraining the model to calculate multiple sets of p-values for combination to produce the result prediction intervals.

Given a training set and test set formulated as $z_1 = (x_1, y_1), \dots, z_l = (x_l, y_l)$ and $x_{l+1}, \dots, x_k$ respectively, the training set $\{z_1, \dots, z_l\}$ we split it into k folds of equal size fitting a model of a combination of k-1 folds and using the final fold as the calibration set and then shifting so that each fold fitted once to a separate model as the calibration set. For each model, we use the same method as split conformal prediction to produce overall k p-values for each test label $(p_1, \dots, p_k)$.

Further variations concern with how the p-values are combined giving a single p-value for each test label. A simple way to combine them is to find its average:

$$\overline{p} = \frac{(p_1, \dots, p_k)}{k}$$

Combining p-values using averaging [21]

However, $\overline{p}$ is stated to not be a p-value itself [21]. Leading to other combination techniques however this is an open problem due to traditional methods such as Fisher's and Bonferroni's assuming dependence or a specific type of dependence which can be violated in a real-world application which is documented with new merging techniques Garsparin et al, [29].

1.2.5 Marginal Coverage and exchangeability

Coverage is the fraction of true class labels that fall within the predicted set for classification and the fraction of true outcomes that fall within the prediction interval given a test set.

$$CCS = \frac{1}{n}\sum_{i=1}^{n} 1\left(y_i \in \hat{C}(x_i)\right)$$

Classification Coverage Score [6],

$$RCS = \frac{1}{n}\sum_{i=1}^{n} 1\left(\hat{y}_i^{low} \leq y_i \leq \hat{y}_i^{up}\right)$$

Regression Coverage Score [6],

These metrics set a confidence value upon the predictions made and add a risk threshold on top of the model to quantify its performance. However, using conformal prediction we can guarantee these coverage scores to be above a set confidence level under the exchangeability assumption.

$$\mathrm{P}\left(Y_{test} \in C(X_{test})\right) \geq 1 - \alpha,$$

Marginal Coverage [7],

With the calibration step as conformal prediction combines the non-conformity scores and user input confidence level to produce the prediction set $C(X_{test})$ for the probability that Y_{test} is in the prediction set this must be above the coverage level for the average over the new points set and is guaranteed through exchangeability.

$$\mathrm{P}(W_1 \in A_1, \ldots, W_n \in A_n) = \mathrm{P}\left(W_{\pi(1)} \in A_1, \ldots, W_{\pi(n)} \in A_{\pi(n)}\right),$$

Exchangeability [9],

Exchangeability is an extremely important assumption that allows for conformal prediction to work, that is, the joint probability of the sequence is unchanged if we shuffle the order of the points. This is weaker than the i.i.d. assumption since exchangeable data need not be independent. Hence the data that has been trained on and the data within the calibration set and test set have the same distribution. Most of the time by simply shuffling before splitting the whole dataset while averaging over trials we can ensure that the marginal coverage we calculate conforms with this exchangeability assumption allowing the calibration phase results of the probability the test set is not contained in the prediction set to be less than equal to alpha which directly corresponds back to the marginal coverage formula. If the distribution changes through bad shuffles and splits or a distribution drift it can cause the marginal coverage guarantee to be flawed preventing the accurate measurement of metrics.

1.3 Bootstrapping

1.3.1 Introduction

Like Conformal prediction, Bootstrapping is a versatile statistical technique that can be applied to non-parametric and parametric models, Classification and Regression problems, and perform validity and efficiency metrics but instead exploits resampling with replacement to generate many statistically important features. Repeatedly drawing samples from the dataset approximates the sampling distribution circumventing the need for strong parametric assumptions and valuably maintains robustness even with small sample sets allowing compatibility with complicated estimation metrics including prediction intervals however traditionally has been thoroughly used with hypothesis testing, bias, variance and other model performance metrics.

1.3.2 Bootstrapping techniques

1.3.2.1 General Structure for prediction

The underlying model will return a point prediction however this isn't very helpful since we don't know to what extent this point prediction is valid or how biased it is. Bootstrapping resamples the dataset to calculate point predictions on the sample fit model and takes a quantile to produce the prediction intervals using the confidence level. A major difference between bootstrap techniques is how the dataset is sampled alongside what is calculated from the sample fit model.

1.3.2.2 Block bootstrap

Block bootstrap acknowledges consecutive data points may be related suiting it to time series domains by sampling sections rather than individual data points resulting in blocks of the original time series structure being maintained however sampling these sections and how they are restitched has led to multiple variations.

Moving Block Bootstrap takes the dataset and uniformly samples with replacement ceiling n/b times between data points ($X_1 \ldots X_{n-b+1}$) where n is the total size of the training set and b is the block size and is restitched by taking the sampled data points and the consecutive b data points and forming a new overlapping sampled training set (truncating the last x datapoints to maintain the same dimension size as the original training set).

Circular Block Bootstrap adapts Moving Block Bootstrap to allow the final b data points to be equally uniformly sampled by connecting the end of the original training data to the beginning meaning if an index from the final b data points is sampled, its corresponding consecutive b data points will continue to the end and continue from the start creating its block. This avoids the edge bias that occurs from Moving Block Bootstrap but depending on the dataset applied this may introduce more problems since the beginning and end of a dataset may look completely different. For example, using a stock, if near the beginning its price is $0.14 and at the end of the training set it reaches $2.50 then datapoints selected from the end will seem that the price suddenly dropped from $2.50 to $0.14 causing a gap in autocorrelation.

1.3.2.3 Sieve bootstrap

Another bootstrap designed for time series is the Autoregressive (AR) Sieve bootstrap that fits a parametric model AR(p) to extract and resample residuals to simulate new bootstrap time series repeatedly increasing the sample size to generate more complicated and flexible AR models capturing its dependence structure non-parametrically to generate bootstrap samples.

Given a sample (x_i, ..., x_n) from a sequence process, we fit an autoregressive model AR(p) dependent on p where p = p(n) -> ∞ and p(n) = o(n) (n -> ∞). Since the AR model is flexible in nature, the function for p generally is an AIC or BIC criteria guiding the model to best fit the data in as simple denominations as possible by penalizing a low log likelihood of models indicating a poorly fit model combined with the number of parameters created for AIC, and penalizing a low log likelihood of the model with a harsher scaling factor for the number of parameters.

$$AIC = -2\log L\,(\hat{\theta}_k) + 2p$$

AIC formula [30],

Smaller AIC scores show a well fit and simple model since we want to increase the negative factor, log likelihood of the model to ensure a well fit model while reducing the positive factor, number of parameters to generate the model.

$$BIC = -2\log L\,(\hat{\theta}_k) + p * \log N$$

BIC formula [30],

N refers to the number of objects in the dataset meaning in many scenarios log N will be above 2 meaning it is slightly harsher towards adding parameters when generating the model. Both methods are suitable to be used for p to guide the AR(p) model.

Coefficients from the AR model treat past lags as individual parameters and once estimated, coefficients from the AR model combined with the sample mean allows residuals to be calculated.

$$\hat{\varepsilon}_{t,n} = \sum_{j=0}^{p(n)} \hat{\phi}_{j,n}\left(X_{t-j} - \overline{X}\right), \quad \hat{\phi}_{0,n} = 1 \quad (t = p+1, \ \ldots, \ n).$$ Residual calculation for Sieve bootstrap [23],

On the originally observed sample, residuals are computed through the summation of the coefficients multiplied by the difference between the value at time t – j and the sample mean. Calculating the residuals alone however do not produce the bootstrap sample since they are merely a measure of error term hence, we need to resample these residuals to produce X*₁, ..., X*ₙ.

$$\tilde{\varepsilon}_{t,n} = \hat{\varepsilon}_{t,n} - (n-p)^{-1} \cdot \sum_{t=p+1}^{n} \hat{\varepsilon}_{t,n} \quad (t = p+1, \ \ldots, \ n)$$

Centring residuals for Sieve bootstrap [23],

However, if the residuals are directly resampled a small bias will be introduced. The residuals are then shifted to have zero mean, centred to remove this bias hence allowing us to generate bootstrapped samples for simulation of future paths in the time series setting of an underlying model and further performative metrics.

1.4 Metrics

1.4.1 Introduction

Understanding the validity and efficiency of a model attaches credibility and certainty to the prediction sets and not just mere point predictions with no context. Both Conformal and Bootstrapping predictors can calculate metrics that state the performance of validity through coverage: Marginal coverage, Conditional coverage – Size stratified coverage, Feature stratified coverage; and performance of prediction efficiency through region size: prediction set size (for classification) and prediction interval width (for regression).

1.4.2 Marginal coverage

As described in the conformal prediction section, this coverage can be guaranteed through the exchangeability assumption and calculates if the fraction true outcomes of the population of cases falls within the prediction interval reaches the desired user input confidence level.

$$P\big(Y_{test} \in C(X_{test})\big) \geq 1 - \alpha,$$

Marginal Coverage [7],

In this formula, Y_{test} is the set of true outcomes and $C(X_{test})$ is the prediction set (Classification), or interval (Regression) given an input set X_{test}, with 1 – alpha equal to the confidence level. If the probability of the set of all true labels is in their corresponding prediction sets is at least equal to the confidence level, then marginal coverage is achieved.

Marginal coverage especially in conformal predictors can be a simple indication the implementation of the predictor is correct since this guarantees true if exchangeability is held and inversely can show if exchangeability has been flouted potentially allowing changes to be made to the implementation of the predictor or the curation of the dataset. In practice, predictors that conform to at minimum marginal coverage are viewed more reliable with an application intended by its implementation. However, as a baseline metric, it doesn't show coverages for sub-groups within neither the input data nor the output data unlike conditional coverages.

1.4.3 Feature Stratified Conditional Coverage

Given a dataset that wants to predict the weight of a person given their age, height and sex, a desirable quality we expect the model to evenly provide correct predictions throughout different ages, heights and sex. Some models may unevenly distribute the correct predictions for example, the model may predict 99% of males correctly however only 91% of females correct given a 95% confidence level hence (provided there is equal female and male in the test set) the marginal coverage would hold however, instinctively we know that the model isn't providing the best predictions for female data. Feature Stratified Coverage illustrates the coverages of each sub-group in each feature demonstrating model weaknesses and although conditional coverage is not possible to guarantee in conformal predictors, methods to improve conditional coverage can adapt existing predictors to provide more even coverage within subgroups.

$$FSC\,metric\!: \min_{g\in\{1,\ldots,G\}} \frac{1}{|\mathrm{I}_g|} \sum_{i\,\in\,\mathrm{I}_g} 1\left\{Y_i^{(val)} \in C\left(X_i^{(val)}\right)\right\}$$

Feature Stratified Coverage metric [13],

Feature stratified coverage splits the test data dependent on its group within a specific feature (i.e. all male test data in one group and all female test data in another) and calculates the fraction of true outcomes in the prediction set, simply its coverage. Min emphasises groups that are significantly lower than the confidence level to express which sub-groups the model is not performing effectively on.

This is especially effective on discrete features since the sub-groups are more obvious however, for continuous data binning the features can cause issues especially when certain demographics are less represented and combined with potentially small sample sets in the case of bootstrapping can cause very skewed overfit results. Equal bin size aims to minimize less represented demographics but inversely merges them potentially making the groups themselves uninterpretable.

1.4.4 Size Stratified Conditional Coverage

Size stratified conditional coverage splits subgroups using calculated prediction sets and specifically their size. For example, if we attempt to predict an image to cat, dog and mouse, Size Stratified Coverage splits the dataset into groups depending on the size of the prediction set (i.e. 1 if the prediction set predicted only cat, dog or mouse, 2 for any 2 from the label set and 3 for all cat, dog and mouse) and calculates the coverage for each group.

$$SSC\,metric\!: \min_{g\in\{1,\ldots,G\}} \frac{1}{|\mathrm{I}_g|} \sum_{i\,\in\,\mathrm{I}_g} 1\left\{Y_i^{(val)} \in C\left(X_i^{(val)}\right)\right\}$$

Size Stratified Coverage metric [13],

Despite being the same formula as Feature Stratified coverage, the meaning of I_g is different since the groups have been split dependent on the size of $C(X_i^{(val)})$ and not the groups inside the features themselves. For regression problems, the prediction interval will need to be binned but causes similar issues as feature stratified coverage for continuous features and similarly for groups that are significantly lower than the confidence level 1 – alpha, suggests inequality and model invalidity.

1.4.5 Region Size

For image prediction, a model can be 100% correct if it predicts all the label options for all test data i.e. if I predicted every digit image to be either 1, 2, 3, 4, 5, 6, 7, 8, 9 or 0 then technically I would be correct. To monitor this and make sure the model produces meaningful results and doesn't blindly guess everything region size metrics illustrate the size of the prediction interval or set. For classification, region size is equal to the cardinality of the prediction set hence $|C(X_i)|$ but for regression, the model predicts a prediction point on a 1d plane, the region size is the difference between the prediction intervals, however many problems require the model to point predict on a multi-dimensional plane hence given 2 vectors of arbitrary dimension, calculate the difference between each plane and multiply by every other plane to produce the region size. Increasing the number of dimensions can cause the curse of dimensionality which can be impractical reaching overflow errors, however using the log region size can mitigate these issues while maintaining information about the size of the prediction intervals. A large region size for a large portion of the prediction intervals may suggest the model itself is not performing efficiently.

2 Implementation and results

2.1 Victor Dheur's multi-output conformal regression [20]

Victor's project has been aimed towards conformal regression problems and contains a clean modular structure structured bottom up, containing existing datasets of regression problems, datamodules that converts the datasets into a useable state to be used by the underlying models (drf-kde, glow, mixture, oracle, quantile and mqf2 are implemented), that can be combined in the conformal predictor phase to generate marginal and conditional coverage metrics alongside region size metrics and some variants. I aim to use the conformal predictor and apply it to a time series setting implementing my own curated datasets exploiting the modular structure to use the datamodules classes, the underlying model mqf2 and the conformal predictor that is implemented and built upon this by independently implementing feature stratified coverage metrics, marginal coverage, size stratified coverage and region size alongside a Moving Block Bootstrapping technique with its calculations for prediction intervals, and hence aforementioned metrics.

2.2 Dataset curation

The stock market provides a typical time series setting for model forecasting and is practically important. Any edge in the stock markets or technique to predict its movements can cause large sums of money to be pulled out, enticing many investors to be interested in any machine learning algorithm that can do so. What's more, risk management is a large part of strategies and is arguably more important than the trade itself and providing confidence levels through Bootstrapping and Conformal prediction for prediction intervals personally interested me than other time series settings. Honestly however, one major reason why I chose this setting is because in my free time I like to play around with trading software and strategies, so bias was a huge factor.

Yfinance is a reliable, free api package that provides financial data from the stock market, forex, commodities and even ETFs from many global markets in real-time. Yfinance also provides a wide variety of time intervals from fast-paced day trading intervals of 1, 2, 5, 15, 30, 60 mins to swing trading intervals such as 1 day to long-term trading intervals of 1 week and even 1 month however in this paper I will only use 1 day intervals again because this is the time frame I am most interested in and swing trading has been said to be the most profitable. I also think this will produce the most reliable results since smaller intervals tend to gap a lot which could suggest weaker correlation between data points and inversely larger time frames could show spikes and anomalous points however if I had more time, I would adjust these time intervals to compare its effect on the prediction intervals between Bootstrapping and conformal predictors.

For the Nvidia data stock I made 2 types of datasets. One containing the log close move for that day as the label and the log close moves for a set shift previous days (10 days) as the features imitating a traditional time series window ensuring autocorrelation between the datapoints through its features and label. The log close move can be precisely calculated as the log of the close on that day minus the log of the close of the previous day:

$$\log c\,lose\,move = \log(Close_i) - \log(Close_{i-1})$$

Choosing the log of the close smoothes the movements between days focusing mainly on the relative changes and limits disproportionate booms or crashes influencing the underlying model predictions in Conformal and Bootstrapping predictors. The curated dataframe is then converted back into a csv file for compatibility of [20] and stored in the data stocks folder for further use and manipulation in the datamodules section.

The other type of dataset contains similarly the log close move as the label but instead uses indicators: K% stochastic, D% stochastic, simple moving average (SMA) 10, SMA 20, SMA 30 and SMA 200, macd, cci, rsi and rsi moving average. Which may be unconventional however again is mainly because of self-interest but uses well founded settings and milestones that are used by many technical traders.

The stochastic oscillator indicator is a momentum indicator that normalizes prices between 0 and 100 as a percentage to compare its closing price with boundaries from the past p days to produce 2 formulas for smooth %K and smooth %D.

$$smooth\,\%K(p,\,ks) = \frac{1}{ks}\sum_{i=0}^{ks-1}\frac{(C_{p-i} - L_{p-i})}{(H_{p-i} - L_{p-i})} \times 100$$

Smooth %K calculation

Given a period, p and a SMA period ks, smooth %K (fast stochastic) takes the SMA over the past ks periods of the percentage of the Closing price of each day – the lowest closing price of the previous p periods divided by the highest closing price of the previous p periods – the lowest.

$$smooth\,\%D(smooth\,\%K,\,ds) = \frac{1}{ds}\sum_{i=0}^{ds-1} smooth\ \%K_i$$

The smooth %D (slow stochastic) is the SMA of period ds of the smooth %K values calculated above. The flexibility in interpretation allows one or a combination of the 2 stochastics to be used in traditional strategies showing divergent-convergent behaviour forecasting both reversal and continuation patterns. Popular inputs for p, ks, ds are 12,3,3.

The Simple moving average is also another momentum indicator that directly builds upon the average of previous prices over a period p, smoothing out fluctuations aiding in general market direction and may serve as powerful support and resistance boundaries which when tested serves the basis of many SMA trading strategies.

$$SMA(p) = \frac{1}{p}\sum_{i\,=\,0}^{p\,-1} Close_i$$

Simple moving average formula

Using the close price of the i^{th} previous day we take the average price over that window. SMA 50 and 200 are popular options with many strategies devised around these indicators.

Moving average convergence/divergence (MACD) invented by Gerald Appel is an indicator comprised of fast and slow exponential moving averages (EMA) generally used to show entry and exit points generally using the 0 line but is known to have many false positives.

$$MACD(fp, sp) = EMA(fp) - EMA(sp)$$

MACD line formula

Given fp as the fast period and sp as the slow period and fp < sp, the difference between the 2 EMA's give the MACD line however, many traders and strategies are implemented based on the histogram which uses a signal line smoothing factor with the previous calculated MACD line.

$$histogram_{MACD}(MACD(fp, sp), sl) = MACD(fp, sp) - EMA_{MACD(fp, sp)}(sl)$$MACD histogram formula

Due to MACDs tendency to generate false positives, it is generally not implemented as a standalone strategy but combined with other popular indicators such as SMA, RSI and CCI to improve robustness.

Commodity Channel Index (CCI) evaluates how far the price has deviated from its historical average identifying areas of overbought and oversold hence providing traders entry and exit points. Despite having no limit between how high or low the value can go i.e. [-∞, ∞] overbought and oversold areas tend to be below/above ± 80 to ± 100 and depending on volatility can go way above these limits.

$$CCI(p) = \frac{TP - SMA_{TP}(p)}{0.015 \times Median\ Deviation}$$

Commodity Channel Index formula [27],

TP is the typical price calculated by averaging the sum of the close, high and low prices of that day. The median deviation is $\frac{1}{N}\sum_{i=1}^{N}\left|TP_i - SMA_{TP_i}(p)\right|$ which is the absolute value of the typical price minus the SMA of the typical price over the period p.

Relative Strength Index (RSI), another popular indicator for technical traders, values between 0 to 100, generally showing overbought areas above 70 and oversold areas above 30 by calculating the average gained over a period divided by the average loss over that same period.

$$RSI(p) = 100 - \left(\frac{100}{1 + \frac{average\ gain_p}{average\ loss_p}}\right)$$

RSI formula [28],

Interestingly, the average gain strictly only adds the total produced by up days divided by p, and the average loss strictly only adds the total produced by down days hence giving the formula above. RSI can also be combined with a smoothing factor to calculate the RSIM providing the EMA over the previous smoothing factor periods.

The combination of these indicators allows many options for the underlying model while providing a suitable time series setting of indicators, however, to maintain no NaN data to be added to the final dataset produced by indicators when calculating moving averages the first 199 datapoints were truncated (since I used SMA200) for the indicators dataset and the first shift + 1 data points were truncated for the log close move dataset.

2.3 Datamodules and preprocessing

Curating the 2 types of datasets containing information over a 5-year period as a csv format allows the reuse and testing carried out through datamodules cleanly loading, processing and storing the dataset into a datamodule class that can be accessed and manipulated by the underlying model pre-implemented by Victor. The datamodules folder has been implemented to extract and preprocess data from many datasets that were provided and the stock datasets that I provided pre-processed by removing rows and columns that were high cardinality, had missing values or pseudo categorical and through a standard scaler after loading and splitting the dataset in training, calibration, testing and validation sets for conformal predictors, and splitting the dataset into training and testing sets for bootstrapping predictors.

In practice for my datasets neither removes any rows or columns in the preprocess provided by Victor. The splitting process is different for Conformal Predictors and Bootstrapping predictors since the Moving Block Bootstrap must maintain its time structure preventing a random shuffle so the predictor can resample with respect to the local time structures given by blocks whereas Conformal Predictors prefer shuffling the entire dataset to generate the required subsets for training, calibration, testing and validation since it can exploit the exchangeability assumption allowing more sections of the dataset to be used for more parts.

After the dataset is split into these subsets, a Standard scaler fits the training dataset to transform the other subsets. Standard scaler works by scaling the features of a data object by subtracting the mean of the features and dividing by the standard deviation:

$$X_{scaled} = \frac{X - \mu}{\sigma}$$

Standard scaler formula [31],

X is set of original feature values applied to all datasets however, the mean μ and the standard deviation σ of the feature are calculated exclusively from the training set. This ensures that the mean of each feature is scaled to 0 and the standard deviation is set to 1 for the training set and uses these calculated values to scale the other datasets accordingly. Despite the datasets most likely not conforming to a normal distribution, the underlying model Mqf^2 uses gradient based optimisation through its Partial ICNN hence standard scaler should assist in its training, converging faster, and improve model performance.

2.4 Underlying model – Mqf2

The underlying model used is the Mqf^2 or multi quantile function forecaster that extends from the univariate quantile function forecasters to capture dependencies downstream modelled using a Partially Input Convex Neural

Network (PICNN) for regression and probabilistic forecasting contexts. Mqf[2] analyses features from previous datapoints, compressing them back into a hidden state through an encoder network PICNN as an output model hence PICNN quantile function is used so its gradient

$$g_\theta(\alpha,\, h) \coloneqq \nabla_\alpha G_\theta(\alpha,\, h)$$

[19]

can model a conditional multivariate quantile function $q_\theta(\alpha \mid h)$ with a quantile vector $\alpha \in (0,\, 1)^n$. Hence exploiting monotonicity guarantees, the multivariate quantile function can be trained on the input features using gradient descent optimizers. The original paper suggests 2 implementations for the training process for which Victor has chosen energy scores. Given a training set with feature output pairs denoted as $\{(x_i,\, z_i)\}_{i=1}^{m}$ he minimizes the energy scores to train the multivariate quantile function:

$$\min_{\theta,\Phi} \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_{ES}\big(q_\theta(\cdot\, |H_\Phi(x_i)), z_i\big),$$

[19]

Which can be extracted to produce point predictions drawn uniformly from the quantile function for use within the conformal predictor or bootstrap methods.

2.5 Conformal Predictor

Desirably we want a predictor that predicts cats as well as it predicts dogs, not trading coverage to predict one class better than the other creating an even environment with reliability and robust properties. Victor's implemented conformal predictor M_CP is a Mondrian conformal predictor that is a variant of split conformal predictors however performs categorical splits to regulate conditional coverages, evenly distributing the errors over the classes.

Given a training sequence Z_{tr} containing (x_1, y_1), ..., (x_n,y_n) where n is the size of the training sequence and a test set containing objects (x_{n+1}, y_{n+1}), ..., (x_k, y_k), we initially split the training sequence into training and calibration sets where Z_t is disjoint with Z_c using a uniformly random split and split sizes of [0.4, 0.1, 0.3, 0.2] for training, validation, calibration and testing datasets respectively. The validation dataset is not used in the implementation of M_CP however a split is still taken since Victor has implemented many other models that do require a validation step. We then train the underlying model, Mqf[2], on the training set Z_t. The adaptation from split conformal resides in this step where a Mondrian taxonomy K divides the calibration set into k disjoint subsets where the categories of the data objects in each individual subset are the same. In the case of classification, the Mondrian taxonomy K can be simply implemented using each unique label as its own split, however we are interested in a regression problem hence this taxonomy is changed to produce bins of equal size for each category. For each category a separate conformal predictor is assigned producing non-conformity scores for values in its calibration set, then a point prediction for a value (x_{n+1}, y_{n+1}) in the test set with corresponding quality estimate $\hat{\sigma}$ allows the category of the test object to be determined producing a list of calibration scores:

$$C_{ji} = \hat{y} + \frac{\hat{\sigma}}{\hat{\sigma}_{ji}}(y_{ji} - \hat{y}_{ji}),\ for\ i \in 1, \dots, q,\ where\ q = |Z_{cj}|$$

List of Calibration scores [32],

These calibration scores take the residuals of the difference between the actual category value and the predicted value and taking a division over the change in quality estimates to calculate a difference measure and adds it back to the predicted value to produce a list of calibration scores.

We can now use a ranking system to generate a scale for the calibration scores for which the predictive distribution can be uniformly sampled using quantile. These predictive distributions will in turn be used to generate prediction intervals for the test object. The process repeats for all other test objects independently to produce a forecasting band.

2.6 Bootstrapping

Moving Block Bootstrapping (MBB) requires the datamodule to initially maintain their local data structure before being resampled so the dataset is split into 2 subsets, training and test [0.7, 0.3], without shuffling the datapoints $Z_{tr} = (x_1, y_1), \dots, (x_t, y_t)$ of consecutive chronological order and test set $Z_{test} = (x_{t+1}, y_{t+1}), \dots, (x_n, y_n)$. Given a block length of size b, the MBB resamples the training data in block chunks to ensure local structure is still represented through a uniformly random selection with replacement of ceil(t/b) data points and attaches its corresponding sequential block chunk. Block chunks are concatenated forming a bootstrap resample and truncated if necessary to maintain the original dimensions of the training set.

A copy of an empty model and trainer is fit using the bootstrap resample and repeated for all bootstrap trials to extract simulated prediction intervals by taking the quantile samples $[q_{\alpha/2}, q_{1-\alpha/2}]$ and computing the efficiency and validity metrics, marginal, feature stratified and size stratified coverages with average region sizes.

2.7 Metrics

Comparing the implemented Bootstrapping and Conformal predictors require metrics to express uncertainty quantification through validation and performance efficiency. Coverages provide validation checks to ensure the predictors and underlying model provide results that aim for a confidence level on test data.

Marginal coverage calculates the fraction of true outcomes from test data that falls within their prediction intervals and if it reaches the desired user input confidence level α.

$$P\big(Y_{test} \in C(X_{test})\big) \geq 1 - \alpha,$$

Marginal Coverage [7],

For conformal predictors this is implemented simply by calculating the prediction intervals for each test object in the test set and checking the fraction of y_{test} is in the prediction interval. For both datasets the label y consists of only 1 value (log close move) forming a regression problem hence testing ∈ in this scenario is computed as: for each $y_i \in Y_{test}$ given the prediction interval $C(x_i)$ where (x_i, y_i) is a test pair, $y_i \in C(X_i)$ is true if $\hat{y}_i^{low} \leq y_i \leq \hat{y}_i^{up}$ meaning the actual value must be higher than the predicted lower bound and less than the predicted upper bound. Marginal coverage for Bootstrapping is computed using the average marginal coverage within each bootstrap trial. Similarly to conformal predictors, the fraction of y_{test} in the prediction interval over the test set size describes the marginal coverage within a single bootstrap trial and then is summed with all other marginal coverages from bootstrap trials, divided by the number of bootstrap trials.

Conditional coverage requires all subgroups of the test set to maintain a high coverage score (larger than the $1 - \alpha$) leading to variations when defining these subgroups through conditions. Feature stratified coverage and size stratified coverage metrics are implemented.

Feature Stratified coverage divides the test data into subgroups dependent on which class of each feature it resides in. For discrete features, we calculate the feature stratified coverage score for each unique value in that feature but for continuous features, we calculate the feature stratified coverage score for each bin of values in the feature. The 3 binning strategies I have implemented for continuous features: Equal width intervals, Equal bin sizes and kmeans. Equal width intervals take a single continuous feature's maximum and minimum difference and divides it by the bin size creating groups to calculate coverage within those groups. Equal bin size orders the X_{test},Y_{test} pair in ascending order of X_{test} and splits it into groups assigned through int(index//(dataset_size/bin_size)) meaning if the bin size is 5 and the dataset size is of 100 test values, each bin will be in increasing groups of 20. The kmeans algorithm can be used as a crude binning strategy to cluster datapoints in the test set allowing independent calculation of coverage within those groups. Using a user input cluster size, kmeans assigns centroids randomly, computing the Euclidean distance between each datapoint to each centroid classifying it into its closest group, and shifts all centroid by calculating its means with their respective cluster datapoints, repeating the process until no shifts to any centroid is made. The coverage is then calculated for each cluster to produce conditional coverage scores.

Size stratified coverage displays validity within groups of prediction intervals. In classification a size stratified coverage score would be measured for each cardinality of the prediction set however for regression, the number of sets would be too large and sparse hence binning splits continuous prediction interval sizes into groups allowing coverage to be measured. The coverage scores are conditioned based on the groups created. For bootstrapping, the feature stratified coverage scores and the size stratified coverage scores are averaged over all bootstrap trials.

Region size tests prediction efficiency for bootstrapping and conformal predictors by calculating the difference between the prediction intervals lower and upper bounds. A histogram displays all the differences showing how large the prediction intervals are and its distribution. This allows predictors to be directly compared since smaller values means a tighter range and can give a more actionable results in the real-world stock example since if the model prediction interval lower bound is still higher than the close of the previous day, then an up trade may be placed whereas if the prediction interval is too large that it cover both bearish and bullish options, it may be harder to develop a strategy that can manipulate the results of the underlying model and predictor.

2.8 Evaluation

Built upon Victor Dheur's multi-output conformal regressionm [20] the contributions I made is adding a Moving Block Bootstrap technique to generate prediction intervals. Calculating marginal coverage, size stratified coverage and feature stratified coverage with equal width binning, equal bin size binning and kmeans binning metrics for both conformal and bootstrap predictors applied to a time series setting through curated datasets from Yahoo finance, yfinance api package on the Nvidia stock to compare both predictors' validity and prediction efficiency.

I will compare Mondrian Conformal (MCP) and Moving Block Bootstrapping (MBB) predictors built upon an underlying Mqf^2 model, carried out twice per dataset (indicator dataset, shifts dataset) for alpha levels 0.1 and 0.2.

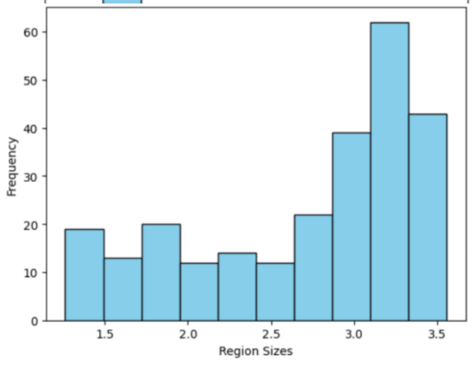
Other variables that have been set stationary are train_validation_calibration_test_ratio for MCP = [0.4,0.1,0.3,0.2]; MBB = [0.7,0,0,0.3]; MBB block_size = 20; MBB bootstrap trials = 2000. For metrics, all feature stratified coverage metrics are initially set to 5 bins (kmeans could reduce this number); size stratified coverage is set to 8 bins; all random functions use a uniform random randomizer not set to any specific seed except for kmeans. 0.1 and 0.2 alpha levels are typical levels for testing time series predictors since higher levels such as 0.05 risk making the prediction intervals too big to compensate for coverage and lower confidence levels trades coverage score leads to reduced accuracy of the prediction especially when coverages may not meet the confidence level.

Despite the training process for Mqf^2 under the conformal prediction method is quick to produce results, since the MBB bootstrapping technique requires a fit of the model per trial, and I have chosen 2000 trials, the time taken to compute metrics can be up to 4 hours on my laptop.

The results below show the MBB and MCP predictors using the indicator dataset at alpha = 0.1.

The marginal coverage for MBB = 0.8867109375 which was surprisingly higher than MCP = 0.8544601202011108, this was quite an unexpected result since traditional conformal predictors tend to have better validity by trading their computational efficiency than bootstrap methods. But this is reversed as we can see from the region sizes below:

MCP region sizes

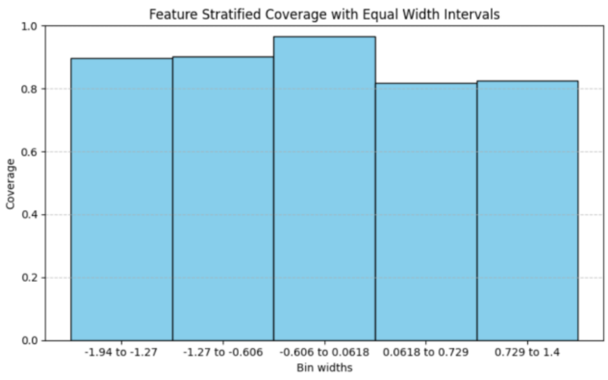


MBB region sizes

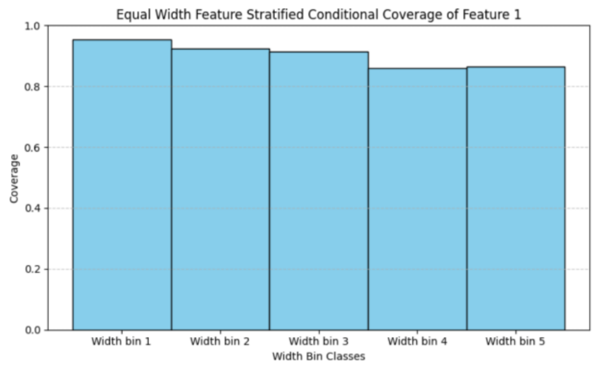
The MCP region sizes are clearly more efficient since many test points have a difference in prediction interval bounds between 1.5 and 2.5 whereas MBB have a large ratio from 2.8 to 3.6, hence producing an interval that is almost double as large. For these MBB test points that produced large region sizes in exchange for 0.03 increase in marginal coverage, it would probably not be worth when implementing a real trading strategy that incorporates these results since the prediction intervals most likely assign the confidence score in both direction therefore we cannot set up an up trade or down trade definitively whereas MCP may be possible to do so or at least has a higher validity.

I think these skewed results is due to the indicator dataset itself since indicators will be using data from previous data points of many days (up to 200 in the case of SMA200) when MBB resamples combining future data points, we have a block size of 20 subsequent data points containing information about the past 200 days potentially confusing the underlying model. We will see a similar trend for alpha = 0.2.

Feature stratified coverage produces a conditional coverage for each individual feature we have in the dataset, which in this scenario is for each indicator. Below shows an example using equal width intervals for feature 1 (%K stochastic):

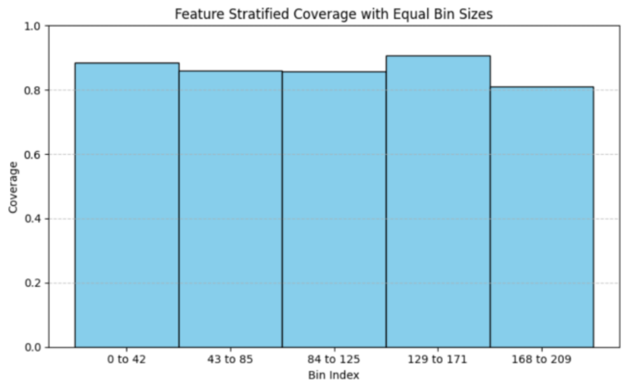


MCP feature stratified coverage using equal width binning for %K stochastic

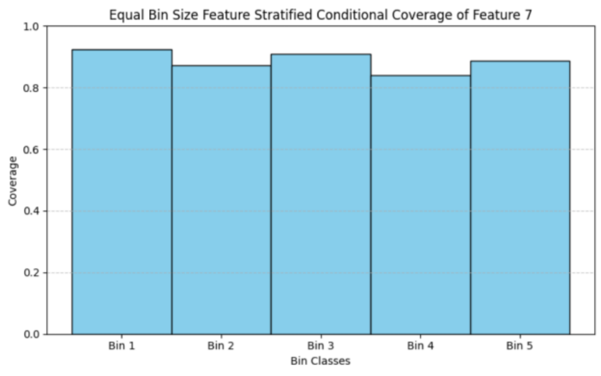


MBB feature stratified coverage using equal width binning for %K stochastic

Despite the bin of each predictor not containing the exact same widths, the overall conditional coverages for %K stochastic are positive with most widths reaching 0.9 (1 – alpha). The last 2 widths of both slightly miss the confidence level however these widths represent increasingly large %K stochastics meaning these may be representing turning points causing a more complex interpretation for the model since it must differentiate if it is showing a bullish or bearish trend. For other features, MBB suffered more in the last width bins of values than MCP sometimes showing coverages lower than 0.4 (MACD feature for MBB) where MCP also struggled but not that extent hence exhibiting higher levels of conditional coverage by MCP. Feature stratified coverage using equal width binning for indicators interestingly showed areas of the indicator that may be more unstable or volatile since it produced dramatically lower coverages. However, when compared to feature stratified equal bin sizes, feature 7 seemed to not be influenced by the upper values potentially showing that there may not be many datapoints from the test set whose feature were in the last bins for equal width binning to allow for sufficient accurate calculation especially since MBB is not shuffling the sequence meaning the trend of the Nvidia stock itself comes into play as the exchangeability assumption is grossly infringed as Nvidia in the dataset starts at a low price ~\$0.14 and is currently around \$170 but has been attempted to be mitigated by taking the log close moves.

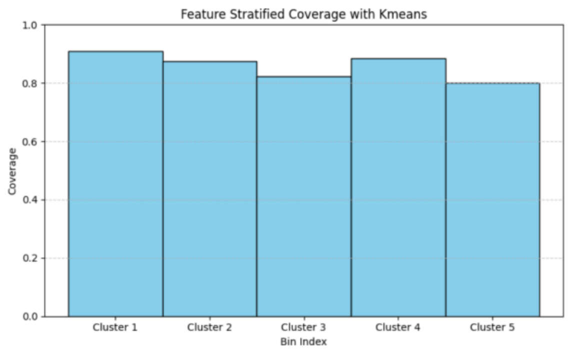


MCP feature stratified coverage using equal bin size for feature 7 (MACD)

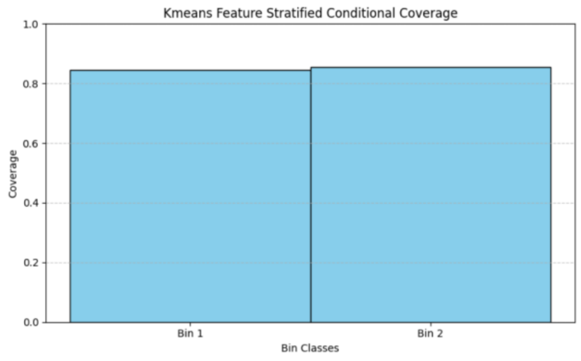


MBB feature stratified coverage using equal bin size for feature 7 (MACD)

The differences between the feature stratified coverage using equal bin size are miniscule for feature 7 between MBB and MCP and equally for all other features. They generally showed similar patterns between bins. It can be noted that equal bin size produced more stable coverages than equal width intervals not producing any major outliers.

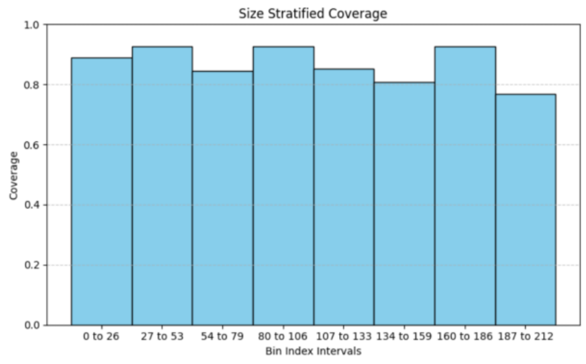


MCP feature stratified coverage with Kmeans

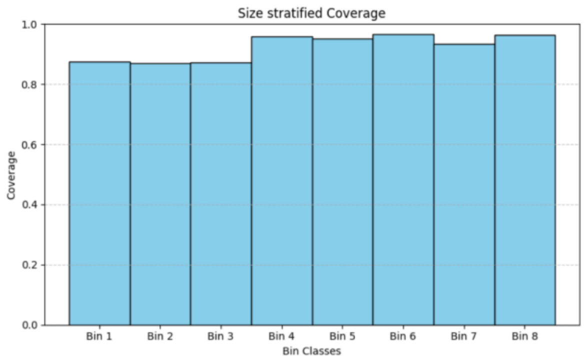


MBB feature stratified coverage with Kmeans

Comparing kmeans for the indicator dataset proves to be challenging due to the MBB feature stratified coverage only produces 2 bins. Despite rerunning this many times, I could never get more than 2 bins unfortunately.



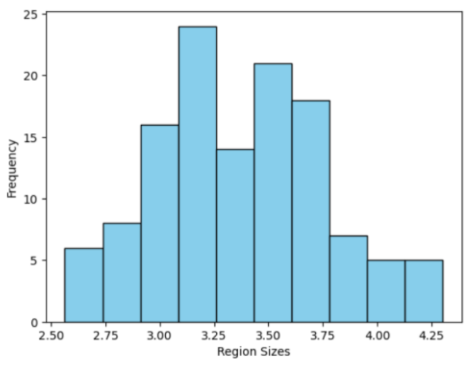
MCP Size stratified coverage



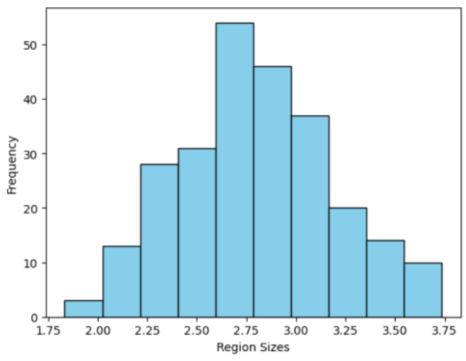
MBB Size stratified coverage

The size stratified coverage for MBB is visibly higher, however, the region size for MBB was much larger than MCP meaning it can a wider net of true values which is most likely the reason for this result. Despite this, the MCP still produced relatively equal coverages except for the last interval.

The results below now show the MBB and MCP predictors using the shift dataset at alpha = 0.1.



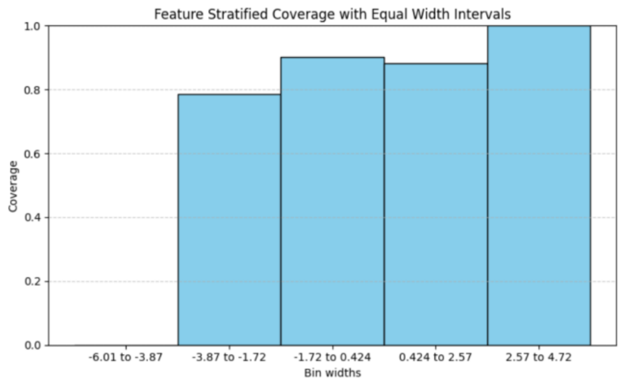
MCP region sizes



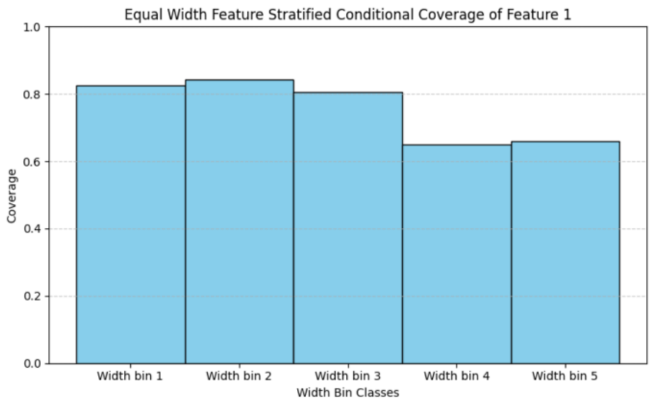
MBB region sizes

As originally anticipated, the region sizes of MCP are slightly larger than the region sizes of MBB region sizes by on average 0.5, however this is compensated by producing more valid prediction intervals calculated by MCP marginal coverage = 0.8799999952316284, compared to MBB marginal coverage = 0.831744140625 which is almost a 0.05 difference which is quite substantial for not a dramatic region size difference.

For feature stratified coverage using equal width binning, many features produced a bin that had 0% coverage.



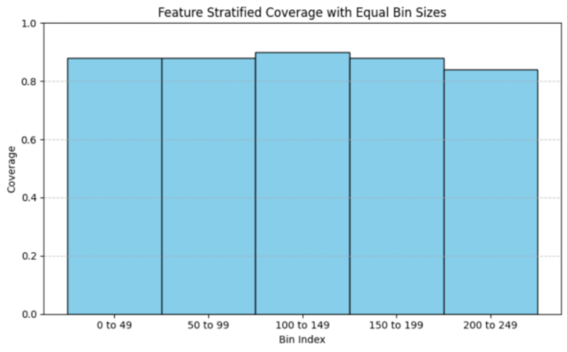
MCP feature stratified coverage using equal width binning for feature 1 (shift 1)



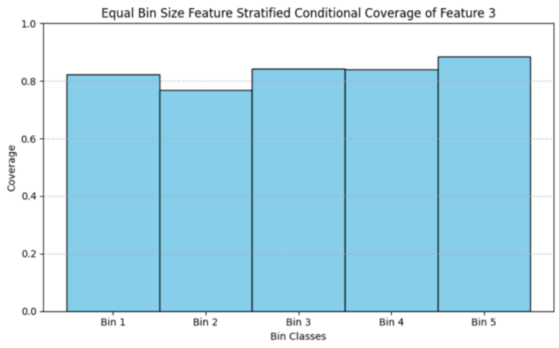
MBB feature stratified coverage using equal width binning for feature 1 (shift 1)

Despite this example showing the issue occurring in MCP, it more often occurred in MBB predictors and is most likely because of the lack of test points that resides

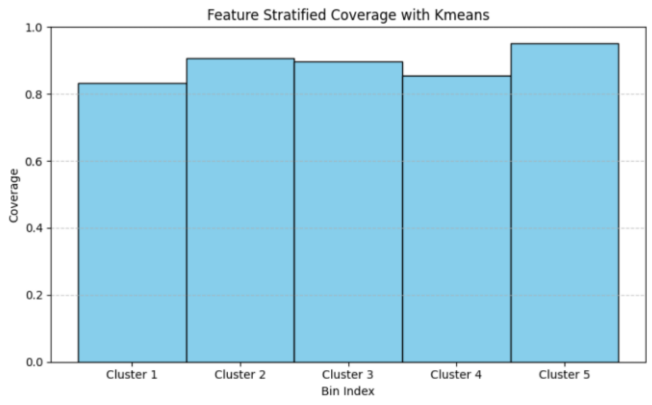
in these bins (i.e. just one or two) meaning the coverage score is highly dependent on whether that single outlier is assigned a prediction point that resides inside the prediction interval, and since they're outliers more often they will not be predicted correctly hence produces these 0%. This is probably an issue with using equal width intervals in general and can be flipped to show a lack of confidence in 1.0 scores since they may also be outliers but happen to be correct.
Despite this, it can still be used to compare both predictors since and generally show that MCP is hitting the 0.9 confidence level in many features whereas MBB misses these coverages by a margin. Feature stratified coverage using equal bin sizes further amplifies the difference between MCP and MBB's validity since MCP not only shows a higher coverage score for the majority of bins for all features, but the variance between bins is minimal which can show that the model is not biased towards a specific condition in each feature.



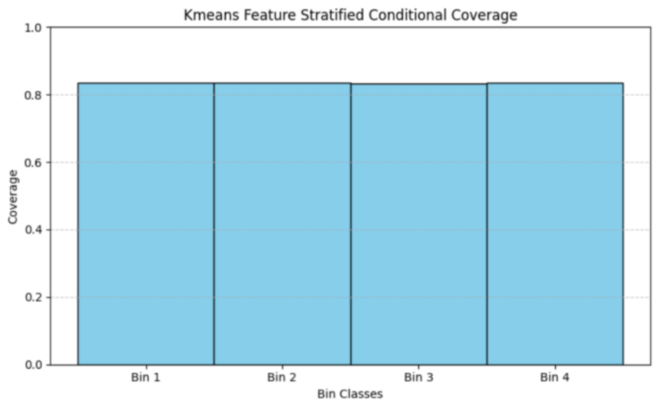
MCP feature stratified coverage with equal bin sizes for feature 3 (shift 3)



MBB feature stratified coverage with equal bin sizes for feature 3 (shift 3)
As you can see, the difference between MBB's highest coverage bin and lowest is quite large which can show some bias and model weaknesses whereas the stability of MCP is ideal despite it not hitting the confidence level consistently (although this should never be sought for and the fact that it reaches close in many scenarios is impressive).

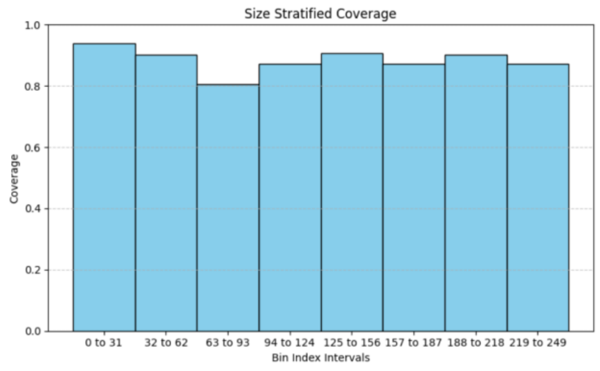


MCP feature stratified coverage with kmeans

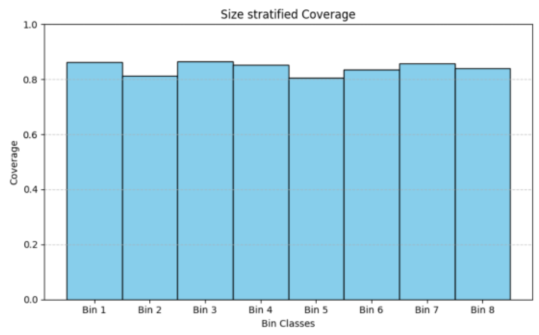


MBB feature stratified coverage with kmeans

Both show relative stability towards their respective bins/clusters however MCP produces higher coverage scores across the range compiling as evidence that MCP is the better prediction framework for this dataset.



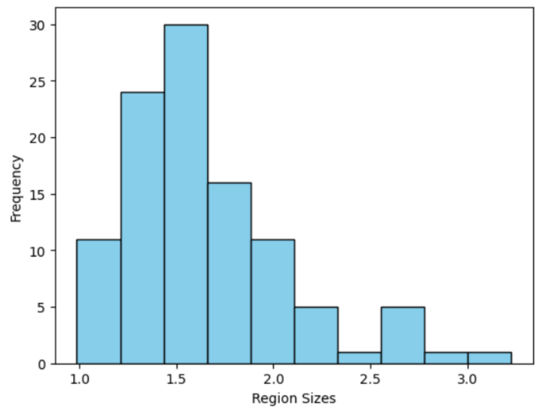
MCP Size stratified coverage



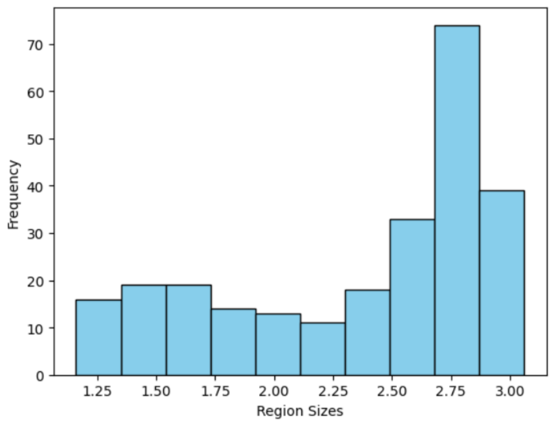
MBB Size stratified Coverage

The region size distributions for both MCP and MBB were similar and hence the size stratified coverage have also been relatively equal. Both predictors have an acceptable level of coverage despite MBB falling short of the confidence level for many bins.

When alpha is changed to 0.2 rather than 0.1, hence leaving more leeway with the validity of the model, everything gets shifted downwards however the distribution hasn't changed much. The region sizes depict this well, comparative to when alpha was 0.1, the region sizes for both MCP and MBB reduce on average, but the distribution looks the same.



MCP region size



MBB region size

This is similarly continued throughout most of the feature stratified coverage scores for equal width binning, equal bin size and kmeans variants alongside size stratified coverage and doesn't really show flaws with either method when working with a lower confidence level. The marginal scores for the indicator dataset (at alpha = 0.2) for MCP was 0.7417840361595154 and MBB = 0.7376328125 And the shift dataset produced marginal scores of MCP = 0.8159999847412109 and MBB = 0.728984375 which is quite interesting since MCP reached its confidence level for the shift dataset even despite the exchangeability assumption not holding whereas MBB didn't even come close severely underperforming potentially worse to when the confidence level was higher.

2.9 Conclusion

This comparative analysis of Mondrian Conformal Prediction (MCP) and Moving Block Bootstrapping (MBB) predictors across two datasets (indicator and shift) at alpha levels of 0.1 and 0.2 reveals nuanced trade-offs between coverage validity and prediction interval efficiency.

While MBB occasionally achieved a higher marginal coverage than MCP, particularly on the indicator dataset at $\alpha = 0.1$, this came at the cost of significantly skewed large prediction intervals. Such inflated region sizes might invalidate practical utility, especially in trading scenarios where directional confidence is crucial.

MCP consistently demonstrated more efficient interval bounds and greater stability across feature stratified and size stratified coverage metrics. Its ability to maintain near-nominal coverage with tighter intervals suggests a more balanced approach to uncertainty quantification. Notably, MCP outperformed MBB in the shift dataset at both alpha levels, even under conditions where the exchangeability assumption was violated highlighting its robustness in real-world time series applications.

Feature stratified coverage analysis further emphasises MCP's superior conditional validity, with fewer bins falling below acceptable coverage thresholds. MBB, in contrast, showed greater variability and susceptibility to outlier influence, particularly in bins with sparse test points and even in bins without sparse test points as shown from equal bin sizes.

Overall, while MBB offers a flexible resampling-based alternative, its computational demands and inflated intervals limit its practicality. MCP is more reliable for predictive uncertainty in this time series setting.

3 Project Diary

Date	Description/goals	Complete
24th June	Initial meeting with supervisor describing early deliverables, research papers and previous implementations to enhance development.	Yes
25th June	Reading through MAPIE implementation of conformal predictors, tsbootstrap implementation and techniques for bootstrapping.	No
26th June	Installation and setup for MAPIE package to see general use cases.	Yes
27th June	Further MAPIE usage to play with a given example to produce coverage metrics.	Yes
3rd July	Recapping MAPIE implementation and reading through some blogs and general information about specific bootstrapping techniques	No, my knowledge of bootstrapping techniques is still questionable
5th July	Reading conformal prediction papers to get a fundamental understanding.	Adequate
8th July	Fortnightly meeting with supervisor, introduction of multi-output conformal regression github from Victor Dheur.	Yes
9th July	Trying to understand github given setting it up on vscode with dependency installations.	No, I am taking the example given to generate preexisting metrics however don't understand what is happening
10th July	Attempting to understand the structure of the code, how it is modularized what specific sections provide.	Yes
14th July	Understanding what specific sections do - looking at the provided datasets where they are stored, what is required to use them.	Yes
17th July	Understanding datamodules section for preprocessing and loading of information from the dataset	Yes, however I don't know how the models use this datamodule.
18th July	Finding how datamodules can be used for the program	No, and I'm getting confused

		by what model and conformal does, their instantiations and using it in the example.
20th July	Finding how datamodules can be used for the program, how model and conformal share information from each other.	No, I'm still confused with runfig parameter passed into the model and conformalizer, what this means and how to apply it.
21st July	Trying to go through datamodules from the example given to better understand datamodules	No, I still haven't understood one bottom-up implementation for a single conformalizer and model pair
22nd July	Fortnightly, meeting with supervisor describing issues with understanding Victor's work. And lack of implementation/ self-produced work.	Yes, I know have plans on what needs to be done given Victor's email to ask about the work.
24th July	Sent an email to Victor about his work, regarding runconfig and which sections I do not understand.	Yes
25th July	Victor replied and many features have become clearer, allowing me to understand how the datamodule splits the dataset allowing usage of the runconfig to hold information about model and conformalizer to be then used for implemented and calculation of metrics	Yes
26th July	I implement my own marginal coverage, researching what is required and coding it onto a separate file	Yes
27th July	Reading a paper including feature stratified, size stratified coverages and region size, with their formulas. Implementing feature stratified coverage.	Partially, I have only started implementing feature stratified coverage since it

		requires binning for the case of continuous metrics.
28th July	Implementing feature stratified coverage using 3 binning techniques.	Partially, I have finished with equal width intervals and equal bin size for binning strategies however haven't started kmeans.
30th July	Kmeans implementation for feature stratified coverage	Yes
1st August	Size stratified coverage implementation and region size implementation	Yes
2nd August	Testing the coverages on given datasets leading to implementing feature stratified coverage for features that are discrete.	Partially, feature stratified for discrete coverage no longer needs bins hence minor adjustments need to be added
3rd August	Adding feature stratified coverage for discrete features and size stratified coverage for classification problems.	Yes, however there were no datasets that can be modelled as a classification problem to generate prediction sets.
5th August	Fortnightly meeting with Supervisor describing implementations of coverage metric for conformal problems and last sprint goals for developing the project.	Yes
6th August	Developing a new dataset using yfinance and implementing indicators for a dataset.	Yes
7th August	Sent an email to the supervisor about the dataset, and adapting it so that I create another different dataset that relies on lags of the past p datapoints	Yes
9th August	Chose a Moving Block Bootstrap as implementation for bootstrapping method and researched what is required. I	Partially, despite my goal to implement a resample of the

	implemented the resampling technique needed.	bootstrap, I realise that I need a new datamodule that extracts data from the dataset and doesn't shuffle the datapoints like done in conformal prediction.
10th August	Implement a new datamodule for Moving Block Bootstrap implementation so that the splits don't reshuffle.	Yes
11th August	Implement a block bootstrap resample	Yes
13th August	Use the resample to carry out a block bootstrap by fitting it in the model.	Partially
14th August	Finishing off the final requirements for the bootstrap technique	Yes
15th August	Calculate block bootstrap region size and size stratified coverage.	Yes
18th August	Testing caches implementation to attempt to speed up metric calculate	No, and despite the training for the bootstrap model over 2000 trials takes 4 hours, I decided to not pursue this further due to time constraints.
19th August	Started collecting technical data for the write up.	Yes
20th August	Fortnightly meeting (one day after) with supervisor describing the results of the project and implementation, what needs to be done in terms of the project report and adjusting the output of the project to produce diagrams for the bootstrapping method.	Yes
21st August onwards	Project report.	Yes

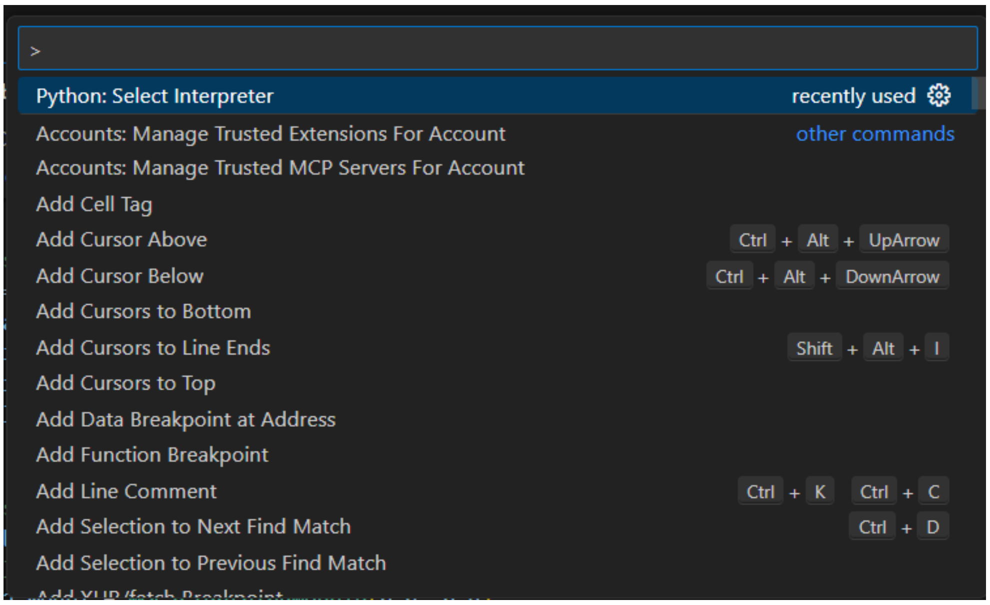
Submission directory structure and running instructions

The submission directory consists of 2 files, the pdf containing this report and the project held as a zip file.

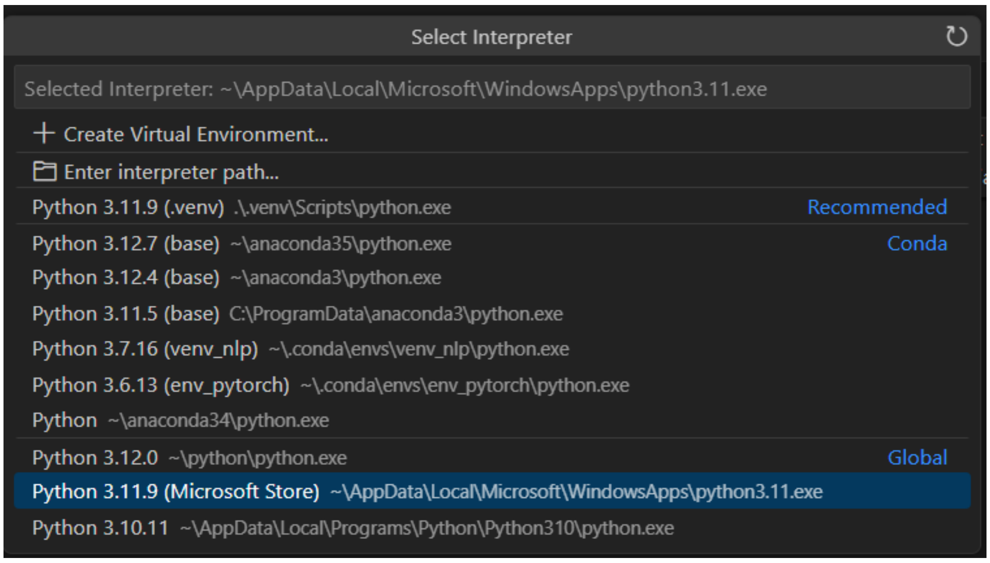
To run the project, you will need the IDE VSCode and a python version (I used 3.11.9 but higher versions should be fine although untested).

Unzip the project folder named 5FINAL_YEAR_PROJECT and add project into your Vscode workspace.

Press ctrl + shift + p to open a tab and click Python: select interpreter



Click create virtual environment and click venv



Select Interpreter

Selected Interpreter: ~\AppData\Local\Microsoft\WindowsApps\python3.11.exe

+ Create Virtual Environment...

Enter interpreter path...

Python 3.11.9 (.venv) .\.venv\Scripts\python.exe Recommended

Python 3.12.7 (base) ~\anaconda35\python.exe Conda

Python 3.12.4 (base) ~\anaconda3\python.exe

Python 3.11.5 (base) C:\ProgramData\anaconda3\python.exe

Python 3.7.16 (venv_nlp) ~\.conda\envs\venv_nlp\python.exe

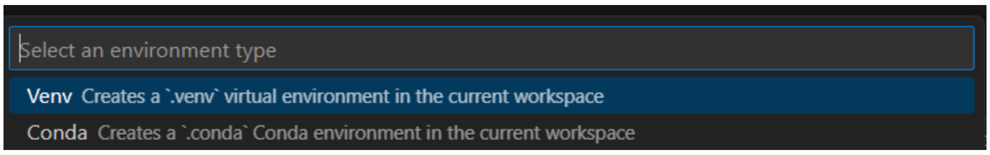
Python 3.6.13 (env_pytorch) ~\.conda\envs\env_pytorch\python.exe

Python ~\anaconda34\python.exe

Python 3.12.0 ~\python\python.exe Global

Python 3.11.9 (Microsoft Store) ~\AppData\Local\Microsoft\WindowsApps\python3.11.exe

Python 3.10.11 ~\AppData\Local\Programs\Python\Python310\python.exe

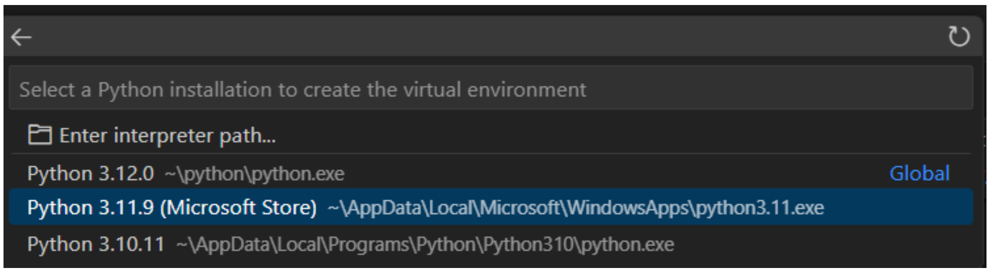


Select an environment type

Venv Creates a `.venv` virtual environment in the current workspace

Conda Creates a `.conda` Conda environment in the current workspace

Then select interpreter of Python 3.11.9 (I have tested only on this version)



Select a Python installation to create the virtual environment

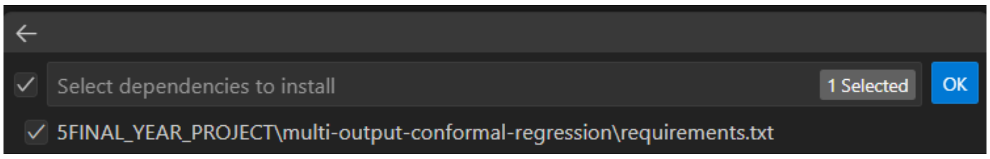
Enter interpreter path...

Python 3.12.0 ~\python\python.exe Global

Python 3.11.9 (Microsoft Store) ~\AppData\Local\Microsoft\WindowsApps\python3.11.exe

Python 3.10.11 ~\AppData\Local\Programs\Python\Python310\python.exe

Then select the dependencies to install and run (approximately 10mins)

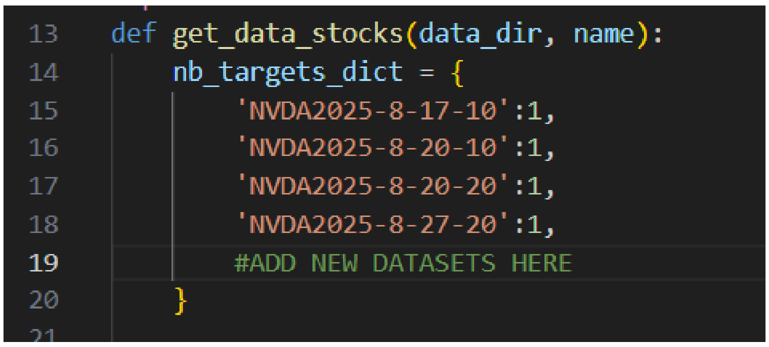


Select dependencies to install 1 Selected OK

✓ 5FINAL_YEAR_PROJECT\multi-output-conformal-regression\requirements.txt

The main run file is the moving_block_bootstrap.ipynb with FinanceData.ipynb as an auxiliary if you want to add a new dataset that has current information of the Nvidia stock (optional).

For FinanceData.ipynb, before running comment out which dataset you want, either shift (log_move_dataset) or indicators dataset and select the jupyter kernel of your dataset and run. The dataset should show in /data/stocks with NVDA followed by the current date and shift. You will need to change shift for the indicator dataset to any other number to not overwrite the shift file despite it not doing anything to its implementation. Then in /moc/datamodules/ for BOTH stocks_datamodule.py and real_datamodule.py you must add the name of the file without csv followed by :1, as shown below:



```
def get_data_stocks(data_dir, name):
    nb_targets_dict = {
        'NVDA2025-8-17-10':1,
        'NVDA2025-8-20-10':1,
        'NVDA2025-8-20-20':1,
        'NVDA2025-8-27-20':1,
        #ADD NEW DATASETS HERE
    }
```

To run the project, navigate to moving_block_bootstrap.ipynb and select the jupyter kernel of venv. Change the dataset to the files you created in FinanceData.ipynb just without the csv. Or use the preset datasets: ‘NVDA2025-8-27-20' indicator dataset, and ‘NVDA2025-8-20-20' for the shift dataset. Then run. Be aware that the first images will be of the conformal predictor which doesn’t take long, and the next set will be of the bootstrap predictor which on my computer roughly takes 4 hours.

References

[1] - V. Vovk, "Transductive Conformal Predictors," 2013. Available: https://inria.hal.science/hal-01459630v1/document

[2] - R. F. Barber and R. J. Tibshirani, "Unifying Different Theories of Conformal Prediction," Available: https://www.stat.berkeley.edu/~ryantibs/papers/unifiedcp.pdf

[3] - Y. Romano, E. Patterson, and E. J. Candès, "Conformalized Quantile Regression," arXiv:1905.03222, 2019. Available: https://arxiv.org/pdf/1905.03222

[4] - A. N. Angelopoulos and S. Bates, "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification," arXiv:2107.07511, 2021, DOI: 10.48550/arXiv.2107.07511. Available: https://arxiv.org/abs/2107.07511

[5] - C. Molnar, "Week #1: Getting Started With Conformal Prediction For Classification," Mindful Modeler, Aug. 2023. Available: https://mindfulmodeler.substack.com/p/week-1-getting-started-with-conformal

[6] - "How to measure conformal prediction performance?" MAPIE Documentation, Available: https://mapie.readthedocs.io/en/latest/theoretical_description_metrics.html

[7] - T. Ding, A. N. Angelopoulos, S. Bates, M. I. Jordan, and R. J. Tibshirani, "Class-Conditional Conformal Prediction with Many Classes," 2023. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/cb931eddd563f8d473c355518ce8601c-Paper-Conference.pdf

[8] - R. F. Barber, E. J. Candès, A. Ramdas, and R. J. Tibshirani, "Conformal prediction beyond exchangeability," Ann. Statist, vol. 51, no. 2, pp. 816–845, Apr. 2023. DOI: 10.1214/23-AOS2276. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-51/issue-2/Conformal-prediction-beyond-exchangeability/10.1214/23-AOS2276.full

[9] - A. K. Kuchibhotla, "Exchangeability, Conformal Prediction, and Rank Tests," arXiv:2005.06095, Jun. 2021. Available: https://arxiv.org/pdf/2005.06095

[10] - R. I. Oliveira, P. Orenstein, T. Ramos, and J. V. Romano, "Split Conformal Prediction and Non-Exchangeable Data," J. Mach. Learn. Res., vol. 25, pp. 1–38, Jul. 2024. Available: https://www.jmlr.org/papers/volume25/23-1553/23-1553.pdf

[11] - V. Vovk, "Cross-conformal predictors," arXiv:1208.0806, Nov. 2024. Available: https://arxiv.org/pdf/1208.0806

[12] - S. Kumar and A. N. Srivistava, “Bootstrap prediction intervals in non-parametric regression with applications to anomaly detection,” Aug. 12–16, 2012. NASA Technical Report, Available: https://ntrs.nasa.gov/citations/20130014367

[13] - A. N. Angelopoulos and S. Bates, “A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification,” Dec. 2022. Available: https://people.eecs.berkeley.edu/~angelopoulos/publications/downloads/gentle_intro_conformal_dfuq.pdf

[14] - B. Radovanov, “A comparison of four different block bootstrap methods,” Croat. Oper. Res. Rev., vol. 5, no. 2, pp. 419–431, 2014. Available: https://ojs.srce.hr/index.php/crorr/article/view/2767

[15] - M. Friedrich, “Sieve bootstrap inference for linear time-varying coefficient models,” J. Econometrics, vol. 236, pp. 1–25, Nov. 2022. DOI: 10.1016/j.jeconom.2022.09.004. Available: https://doi.org/10.1016/j.jeconom.2022.09.004

[16] - D. Politis and J. Romano, “Recent Developments in Bootstrapping Time Series,” J. Econometrics, vol. 91, no. 1, pp. 75–104, Jan. 1999. DOI: 10.1080/07474930008800457. Available: https://www.tandfonline.com/doi/pdf/10.1080/07474930008800457

[17] - “API Documentation — yahoo finance documentation,” python-yahoofinance.readthedocs.io, Available: https://python-yahoofinance.readthedocs.io/en/latest/api.html

[18] - G. Bland, “yfinance Library - A Complete Guide” Quantitative Trading Ideas and Guides - AlgoTrading101 Blog, Jun. 16, 2020. https://algotrading101.com/learn/yfinance-guide

[19] - K. Kan et al.,“Multivariate Quantile Function Forecaster,” arXiv:2202.11316, Feb. 2022. Available: https://arxiv.org/pdf/2202.11316

[20] - Victor Dheur, “Multi-output conformal regression,” Available: https://github.com/Vekteur/multi-output-conformal-regression

[21] - V. Vovk and R. Wang, “Combining p-values via averaging,” arXiv:1212.4966, Oct. 2019. Available: https://arxiv.org/pdf/1212.4966

[22] - A. M. Alonso, D. Peña, J. Romo, “Forecasting time series with sieve bootstrap,” Jan. 2002. Available: https://core.ac.uk/download/pdf/29403068.pdf

[23] - P. Bühlmann, “Sieve Bootstrap for time series”, Bernoulli, Vol. 3, No. 2, pp. 123-148, Jun. 1997. DOI: 10.2307/3318584, Available: https://www.jstor.org/stable/3318584

[24] - “Stochastic oscillator,” Wikipedia. Available: https://en.wikipedia.org/wiki/Stochastic_oscillator

[25] - “Stochastic Oscillator (STOCH),” Trading Technologies, 2025. Available: https://library.tradingtechnologies.com/trade/chrt-ti-stochastic-oscillator.html

[26] - A. Hayes, “Simple Moving Average - SMA,” Investopedia, 2024. Available: https://www.investopedia.com/terms/s/sma.asp

[27] - J. Kuepper, “Timing Trades With the Commodity Channel Index,” Investopedia, 2018. Available: https://www.investopedia.com/investing/timing-trades-with-commodity-channel-index

[28] - J. Fernando, “Relative Strength Index (RSI) Indicator Explained with Formula,” Investopedia, Nov. 19, 2024. Available: https://www.investopedia.com/terms/r/rsi.asp

[29] - M. Gasparin, R. Wang, and A. Ramdas, “Combining Exchangeable p-values,” arXiv:2404.03484, Feb. 2025. Available: https://arxiv.org/pdf/2404.03484

[30] - G. H. Lubke et al., “Assessing Model Selection Uncertainty Using a Bootstrap Approach: An update”, 2017. DOI: 10.1080/10705511.2016.1252265 Available: https://doi.org/10.1080/10705511.2016.1252265

[31] - “StandardScaler, MinMaxScaler and RobustScaler techniques ML,” GeeksforGeeks, Jul. 15, 2020. Available: https://www.geeksforgeeks.org/machine-learning/standardscaler-minmaxscaler-and-robustscaler-techniques-ml

[32] - H. Boström, U. Johansson, and T. Löfström, “Mondrian Conformal Predictive Distributions,” Proc. Mach. Learn. Res., vol. 152, pp. 1–15, 2021. Available: https://proceedings.mlr.press/v152/bostrom21a/bostrom21a.pdf