Forecasting Urban Indicators with Machine Learning Methods

Michael Freeman

Supervisors: Riccardo Di Clemente and Bernardo Monechi

April 2022

Abstract

Urban Scaling Laws, the study of how observations of a city scale with the size of the city, has become a popular and important area of research in city science. By deriving new statistics from these laws, statistical models have been used to model how cities evolve over time. We opt instead to use machine learning methods. We find that US counties also follow the scaling laws that have been previously formalised for cities. After applying the rescaling the majority of the instances in the dataset were seen to be stationary. We overcome this problem by using a Dynamic Time Warping based clustering approach to extract the non-stationary points. Linear regression is found to be an effective general model for predicting the rescaled urban indicators. However our experiments find that neither linear nor ensemble based approaches can be used for a general model for the derivatives of the data.

I certify that all material in this dissertation which is not my own work has been identified. Michael Freeman

Contents

1 Introduction

The science of cities and how they interact and grow is an important area of study. This allows the prediction of crime [1,2], predicting the economic aspects of cities [3,4] or the improvement of the efficiency of cities [5]. In recent years as data has become more readily available and in sufficient quantity there has been an increased interest in city science and how it can be used. With the UN estimating that 55% of the global population live in cities, and with that figure set to rise to 68% by 2050 [6], this field of research is only going to become more important

Urban scaling laws is an area that has recently become popular in this field. This is the study of how observations of a city scale with the size of the city. Formally a scaling law of a function Y of a variable X is described by the equation

$$\frac{Y(\lambda X)}{Y(X)} = f(\lambda) \tag{1}$$

where λ is a scaling factor and f is a function of λ independent of X. In fact the idea of urban quantities scaling with the size of a city is a natural concept. Many observations of a population are often reported more as rates rather than the original statistic, for example income and GDP are often reported per capita and crime is usually reported as a rate per 100,000. However, these measures assume that the quantities grow linearly with the population which may not be the case, it may therefore be interesting to investigate whether there is a non-linear scaling between urban quantities and population. This has been the subject of much research. If y is the observation of the city and x the population of the city, a celebrated result is that urban scaling laws follow a power law

$$y = \alpha x^{\beta} \tag{2}$$

When we have a $\beta < 1$ we call the relation sub-linear, and super-linear if $\beta > 1$. It is often observed that if y is a creative or socioeconomic observation, such as patents filed or GDP, there is a super-linear scaling, whereas for infrastructure observations a sub-linear scaling is observed [7–10].

Using statistics derived from these power laws there has been some research into using statistical models to predict these derived statistics as they evolve over time [3, 11]. However due to the complexity of cities and therefore the complexity of the relationships between different parameters, it may be more suitable to use machine learning methods to make predictions. This is an area where there has been very little research done, yet may prove fruitful, therefore is important to investigate. In this project we investigate how well machine learning methods perform the task of forecasting how urban indicators change in time.

2 Literature Review

In an early investigation into Urban Scaling Laws, Bettencourt *et al.* [9] investigate scaling laws for socioeconomic activities. They argue that linear scaling is unsuitable for such phenomena as it ignores the non-linear interactions that occur due to social

dynamics. As a motivating example for this, they observe that as the population of a city doubles, the economic productivity increases by $\approx 15\%$, regardless of the starting population, indicating a scaling law with $\beta \approx 1.2$. In order to better compare and rank different cities based on an urban characteristic, they introduce Scale Adjusted Metropolitan Indicators (SAMIs) defined for a city i by

$$\xi_i \coloneqq \log\left(\frac{y_i}{\alpha x_i^{\beta}}\right) = \log\left(y_i\right) - \log\left(\alpha x_i^{\beta}\right) \tag{3}$$

where y_i is the observed value of the characteristic of the city, x_i is the city's population and αx_i^{β} is the estimate for y_i using the urban scaling power law. They suggest that this is a better measure than per capita measures as they are free of dimension and independent of city size and other urban metrics. They argue and show that using SAMIs shows less bias towards larger populations than per capita measures, when urban scaling fits well. They also find that the distribution of the ξs is modelled well by a Laplacian Distribution, which suggests that the statistics of urban indicators also follow scaling laws. This approach also allows Bettencourt *et al.* to quantify how much a city is improving over time by computing the autocorrelation between measures taken at different times.

In a further paper [8], Bettencourt *et al.* formalise urban scaling laws, allowing the use of a power law (Equation 2), and discuss certain implications and difficulties that arise from using scaling laws. They argue that a problem with the use of a power law is the range of data that is available. While in many biological or physical systems where a power law may be used, the data has a vast range, the data for cities is limited. In the available data the largest populations are in the order of 10^7 while the smallest tends to be in the order of $10^3 - 10^4$. This gives a rather short range from which to draw conclusions about the distributions of data.

Much of the literature regarding urban scaling laws focuses on how an urban indicator changes as population does, however, recent studies [3, 11] have considered time as well as population to be an important variable for scaling analysis. Bettencourt *et al.* [11], consider two different approaches to scaling analysis *cross-sectional* and *temporal* scaling. Cross-sectional scaling keeps time constant and compares the properties of all cities in a system. While temporal takes the evolution of a certain quantity of a specific city in time and fits it to a power law of its population as this changes over time. They then argue that with some simplification and ideal circumstances that the cross-sectional and temporal cases are equivalent. However they note that for cities with very low population growth, as is increasingly the case globally, the deviation between the models becomes large. Their results for temporal analysis on congestion costs for US urban areas, show that that while the majority of areas have a similar scaling exponent of around 2, the extremes are vastly different reaching values of 10 and -32. They suggest this is caused by a low or negative population growth in post-industrial cities causing a divergence. These results also show that for some cities the model did not fit the data well, suggesting temporal fits may be rejected. This seems to suggest that a scaling law alone is not sufficient to forecast the evolution of cities.

Monechi *et al.* [3], propose a different solution to this, that does allow for prediction of the future state of a city, based on a Maximum Entropy (ME) approach. They assume that a scaling law is an intrinsic part of the dynamics of a city and use this to build their model. Monechi *et al.* rescale the indicators to be the SAMI (Equation 3) of the indicator scaled by the standard deviation of the indicator. This then translates the value for each instance to be how many standard deviations away from the expected value given a power law scaling each indicator is. For each urban area they build a vector $\mathbf{x} = (x_i)_{i=1}^{N}$, where each x_i is a rescaled indicator. They then build a generative probability distribution $\mathcal{P} : \mathbb{R}^N \to \mathbb{R}$, using an energy based probability distribution. testing their model they find that only a small percentage ($< 2\%$) are incompatible. To consider the dynamics of the vectors $\mathbf{x}$ in time Monechi *et al.* use a Langevin equation [12]. They find that there is strong agreement between the data and Langevin model for a difference between consecutive times, validating the use of a Langevin model. Despite this there is less agreement between the variations of the data and model. However, they find that if they restrict the dataset to only urban areas with a population $> 10^4$, there is a greater agreement between the model and data. This would the suggest that the model is good model for forecasting large cities.

2.1 Project Specification

In this project we investigate how machine learning methods can be used to forecast urban indicators in time. This is similar to the work of Monechi *et al.* [3], who use statistical models rather than machine learning methods. Should we obtain good enough results we compare these to those of Monechi *et al.* . This requires us to process and analyse a dataset of urban indicators before then applying machine learning methods to it in order to obtain predictive models. We will always first fit a linear regression model as this is the simplest model and if this receives good results there is no point using more complex models. Should a linear regression fail, we use ensemble learning methods. This is due to the fact that they greatly decrease the variance of prediction when compared to their base regressors such as regression trees. The specific ensemble methods we use are random forests and gradient boosting. We choose these methods as random forests are being used for econometric data increasingly frequently [13] and there is some evidence to suggest that in some applications gradient boosting and its variants achieve better results than random forests [14, 15].

To achieve this we use python for our analyses. This is due to the fact that it has many packages for data science and machine learning, such as Pandas, SciKit-Learn [16] and matplotlib. All machine learning methods used are implementations from SciKit-Learn, except for §4.2.2 where we use Tslearn [17], a similar package designed for time series data. All code is provided in jupyter notebooks.

3 Data

For this study, the data that we use must include both urban indicators as well as population for the urban area, so that we may rescale the indicators to their SAMIs

(Equation 3) [8]. There are a number of sources of data that we could use with this property each with their own advantages and disadvantages.

3.1 OECD (Organisation for Economic Co-operation and Development)

The OECD (Organisation for Economic Co-operation and Development) is an intergovernmental organisation created to stimulate world trade and economic progress, with 38 member countries around the world. They produce a database of Metropolitan data of cities from its member countries [18]. This includes data on crime, economy, income and employment as well as the population of 691 Functional Urban Areas (FUAs). An advantage of using this dataset is that each FUA has a population of at least 10^5, this proves useful as Monechi *et al.* [3] find that their model works best for populations of at least 10^4. The database contains annual data from 2000 to 2021, however each FUA doesn't necessarily contain data for each feature every year or at all. Therefore, after processing the data we obtain very short periods for learning and each dataset containing very few instances. Together this also means that it is very difficult to impossible to create a dataset of the combined features. Due to these problems, this dataset is unsuitable for applying machine learning methods.

3.2 U.S. Counties

The U.S. Bureau of Labor Statistics produces Quarterly Census of Employment and Wages (QCEW) [19] data for the 3,143 US counties and 78 municipalities in Puerto Rico. As the name suggests this contains quarterly data on wages and employment, as well as aggregating this at an annual level. For each county this dataset reports the number of establishments, the number of people in employment and the total wages earnt, which will call *estab*, *emp* and *wage* respectively from here.

While this dataset contains urban indicators, it doesn't contain the population that we need to calculate the SAMIs (Equation 3). For this we use the population data from the US Census Bureau [20, 21]. These report the population for each county annually. Population, however, is not a feature that is recorded annually, but rather every 10 years with the census, therefore for intercensal years the reported population is an estimation.

Despite the QCEW data being reported quarterly, we will not be using the quarterly resolution but rather the annual. This is for a number of reasons, the first of which is the noisy nature of the quarterly data. The noise makes it difficult for machine learning methods to fit to. Fortunately, however, we find this noise cancels itself out when the data is collated on an annual level. Another reason for not using the quarterly data is due to the fact that the population is reported annually rather than quarterly and we would need to interpolate this to obtain quarterly data. Since for many years in the dataset the population is already an interpolation, further interpolating may lead to inaccurate results.

A problem that we may encounter with this dataset is that it uses counties rather than cities. Counties are, in many cases, far more sparsely populated than cities,

which means that it is unlikely that the standard dynamics of cities will apply here. This may then lead to standard scaling law (Equation 1) not applying well, however as we see in §4.1 this is not the case.

3.3 French Communes

The final dataset that could be used is for French communes for years between 2006 and 2015 from the INSEE (the French Institut National de la Statistique et des Études Économiques) [22]. Communes are the smallest administrative division in France which contains close to 35,000, making this by far the largest of the three datasets. This dataset contains data on many socioeconomic factors such as the number of individuals working in certain sectors as well as the employment rate and rate of highly educated people. This is also the data that Monechi *et al.* use in their study [3] and they provide their processed datasets [23]. Therefore using this dataset would give a good way to compare the machine learning methods I use to the current statistical methods.

We decide that for this study we will use the US QCEW data from the US Bureau of Labor Statistics. This is due to the fact that it provides a large enough set for building meaningful predictive models on with machine learning methods, unlike the OECD data. It will also be interesting to investigate how well larger and less densely populated areas such as US counties follow the power law (Equation 2) which has been formalised for cities.

4 Data Processing

4.1 Computing SAMIs

Before carrying out any investigations we must first need to process the data so that it is in the format that we wish. Following Monechi *et al.* [3], we rescale the urban indicators to their SAMIs (Equation 3). In order to do this we must first assume there is a power law relating the urban indicator, y, and the population, x, as in Equation 2

$$y = \alpha x^{\beta}.$$

We then need to find the scaling coefficients α and β. While at first this may seem like a difficult non-linear problem, we may reduce it to a linear problem by taking logs giving us

$$\log y = \log \alpha + \beta \log x. \tag{4}$$

We may then find the best α and β by fitting a linear regression to this using our data. A problem that arises when using this method, particularly with the US QCEW data [19], is that since some US counties are quite small there may not be any wages earnt, establishments open or employed members of the population, i.e. in counties with a large retired population. Therefore the value of the indicator y will be 0 and when taking the log of this we get $-\infty$, a value that is not useful for our purposes. To remedy this we add 1 to all urban indicators. Since we may expect some natural

annual variation in the scaling coefficients, we calculate separate scaling coefficients for each year. To remove any outlier urban areas after finding the set of scaling coefficients, we calculate the SAMIs and remove any urban area with a SAMI from any year that lies at least three standard deviations away from the mean. We then recalculate the scaling coefficients to remove the effect the outliers would have on the fit of the linear regression, the effect of this can be seen in Figures 1a and 1b. To standardise the SAMIs across the years we also divide them by each years' standard deviation. We omit subtracting the mean prior to dividing by the standard deviation as due to the nature of how SAMIs are constructed their mean will be 0 or close to 0. Combining the SAMIs from each year we can see in Figure 1d that we get a distribution close to that of a Gaussian.

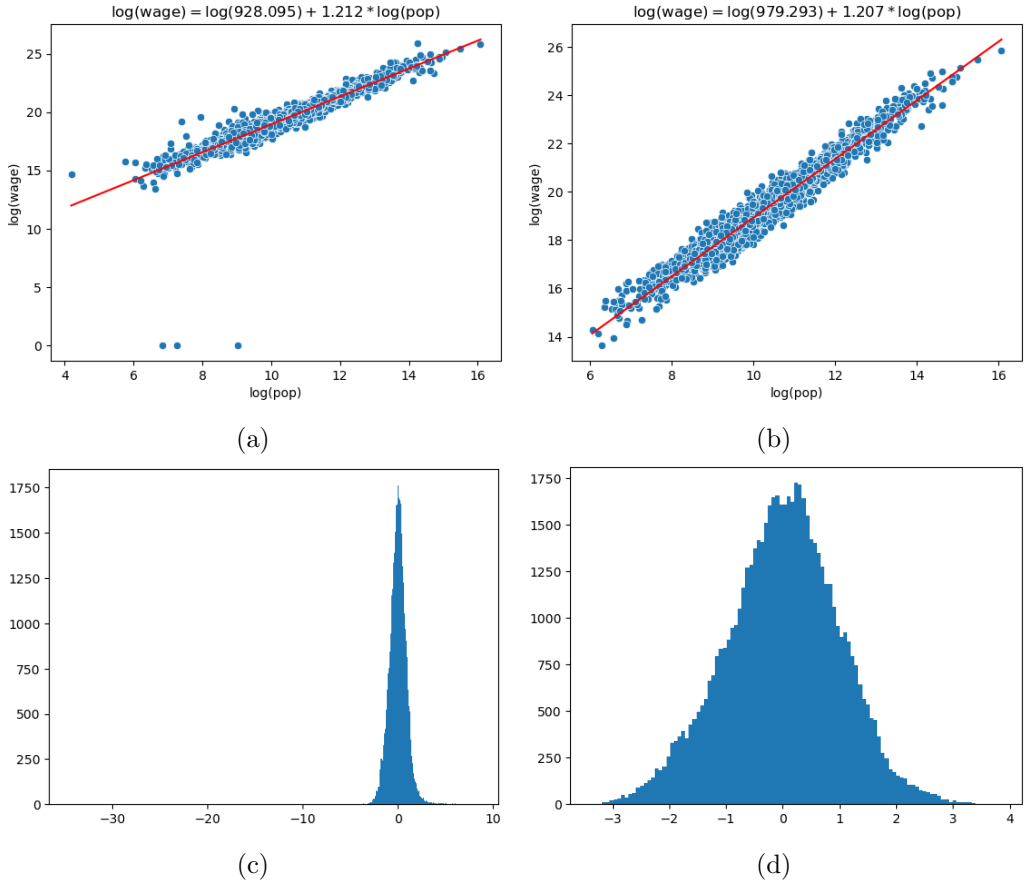
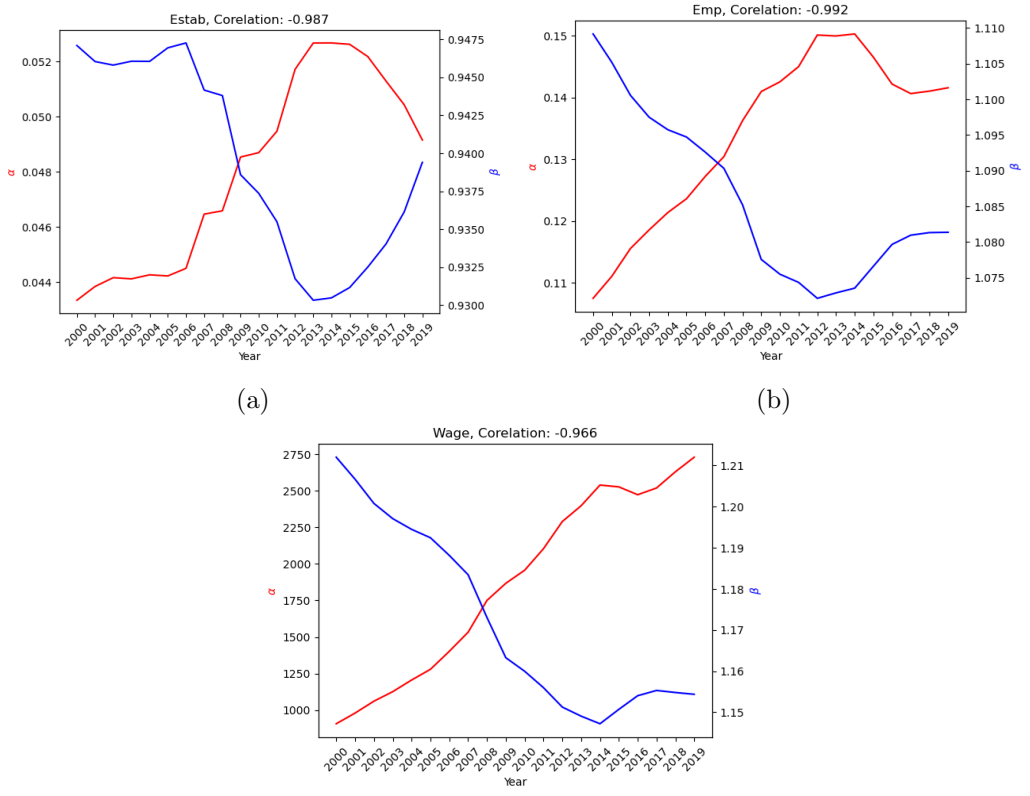


Figure 1: (1a) Fitting of linear regression on wage data from 2001 before removing the anomalous counties. We can see these anomalous counties are those close to the x-axis and those straying further from the line of best fit in red. (1b) The fitting of the linear regression to thew log plot after removing the anomalous counties. We can see that this is a better fitting model than that in 1a. (1c) Histogram of the SAMIs for wage for all years before the removal of anomalous counties. We see there at the anomalous counties have SAMIs in the neighbourhood of -30, these correspond to those points in 1a close to the x-axis. (1d) The distribution of SAMIs for Wage for all years. We obtain a distribution close to a Gaussian.

Analysing the scaling coefficients we get over time, we notice a relationship between α and β. This is shown in Figure 2. We notice that in all three cases α and

β are almost inversely proportional. This is supported by the correlation coefficients, all strongly negative, -0.98, -0.99 and -0.97 for *estab*, *emp*, and *wage* respectively. Due to this relationship I will be referring only to β for the remainder of this discussion. For all three indicators we see that the value of β falls in the early 2000s, however we see that the rate of decrease increases around 2008. β continues to fall until 2012-2014 when we see the value of β begin to rise again. This pattern seems to match the 2008 financial crisis [24] and some estimates [25, 26] of when the economy began to recover from this. Beyond this we see that the βs for *emp* and *wage* seem to stabilise after 2016, while it continues increasing for *estab*. Another pattern that emerges is that the βs for *emp* and *wage*, seem to be quite closely correlated. This would make sense as we would expect that having a larger portion of the population in work will lead to a larger amount of wages being earnt.



(a) (b)

(c)

Figure 2: The relationship of the scaling coefficients α and β over time. We see that there is a strong negative correlation between the two and that their rates of change reflect the 2008 financial crisis and its recovery.

4.2 Clustering

Now that we have rescaled the urban indicators to their SAMIs, we are in theory ready to start applying some prediction models. However we note that a vast majority of the time series are stationary. That is to say if a county has a SAMI 1 standard deviation higher than the average in year t, it is likely to have a SAMI

close to 1 standard deviation higher than the mean in year $t+1$. This can be illustrated by fitting a linear regression model to the data. If we take data from years $t-n,\ t-(n-1),\ldots,\ t-1,\ t$ to predict year $t+1$ we obtain a linear regression with coefficients close to 0 for years $t-n,\ t-(n-1),\ldots,\ t-1$ and a coefficient close to 1 for year t, i.e. the model is just producing the SAMI from year t for year $t+1$. We get a coefficient of determination of 0.98 on a 20% hold out test set with this model, suggesting that the model where we just predict the previous year fits the data very well. This fact is illustrated in Figure 3, particularly in the second half of the time series. Were this the case for all of the data it would not make sense to build any more sophisticated models to predict the SAMIs. Therefore we need to be able to extract the non-stationary time series from the stationary. To do this we can use clustering.

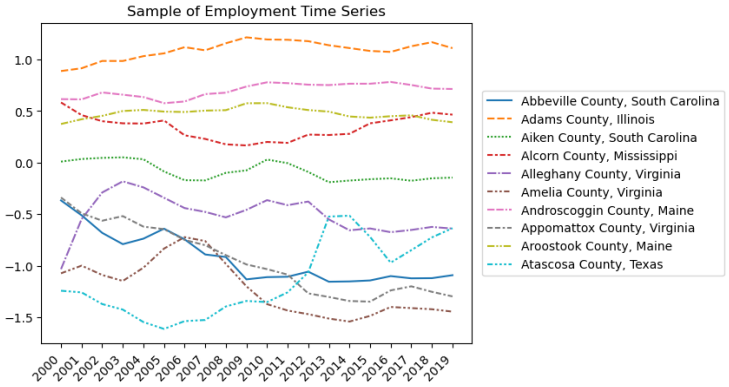


Figure 3: A sample of the dataset for employment. We can see that a large number of the time series are stationary, that is they are horizontal lines. This is particularly prominent in the second half of the time series, after 2011.

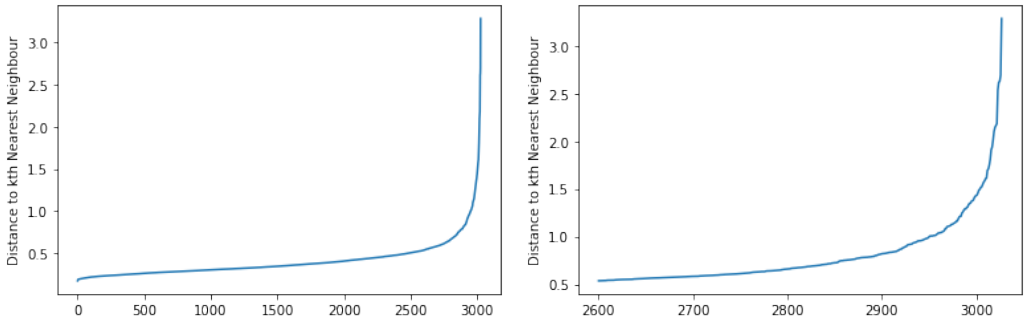
Clustering is a class of unsupervised machine learning methods used to separate a dataset into various groups based on some similarity metric. This can used to find hidden patterns in data and is therefore useful for our purposes as we wish to identify points in our dataset that are non-stationary over time. The most common method of clustering, k-means, clusters data-points based on their distance as points in Euclidean space, however we wish to cluster points based on the shape of their time series, therefore we must use a different method.

Since we are aiming to cluster the points based on their shape or where their time series goes in 2D space, i.e. its trajectory, we need to use a method for these purposes. There has been some research in recent years into clustering based on trajectories [27–29], which has applications in areas such as object tracking. The methods in this research, however, are often either supervised or semi-supervised, depending on having actual groupings for some or all of the data, and focuses on finding representatives for each cluster of trajectories. One thing that these methods do have in common though is that they use DBSCAN as part of their methodologies, hence we start by using this method to try to cluster our data.

4.2.1 DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) [30] is a clustering method that based on the density of the points nearby. It assumes that clusters are areas in space consisting of densely packed points, separated by areas of points with low density. The density of space around a point p_0 is computed by counting the points in an ϵ-neighbourhood of p_0, that is counting the points p such that $|p_0 - p| < \epsilon$ for some predefined $\epsilon > 0$. A point p_0 is then said to be a core point if for some predefined $minpts > 0$, $|\{p : |p - p_0| < \epsilon\}| > minpts$. If a point p is not a core point but there exists a core point p_c, such that $|p - p_c| < \epsilon$, then p is a border point, otherwise p is considered to be a noise point. Any two core or border points, p, q, are considered to be in the same cluster if there is a chain of points $p = p_0, p_1, \ldots, p_{n-1}, p_n = q$, such that $|p_{i-1} - p_i| < \epsilon$ for all $i = 1, \ldots, n$. The DBSCAN algorithm works by first scanning over the points in the dataset and classifies them as either core, border or noise. It then assigns core and border points to different clusters using the rule mentioned previously.

There are two important hyper-parameters in this algorithm that need to be tuned to obtain optimal results, ϵ and *minpts*. There are suggested methods for finding optimal values for these hyper-parameters [31–33]. For n-dimensional data, it is suggested that an optimal choice for *minpts* is $2n$. This means that for our purposes, where we are viewing a 20 year time-series dataset as set of points in 20 dimensional Euclidean space, we should take *minpts* to be 40. The method for determining ϵ is slightly more complex and relies on our choice of *minpts*. To estimate an optimal ϵ, we first fit a K-Nearest-Neighbour model to the data with $k = minpts$ and from each point consider the distance to the k^{th} nearest neighbour. We then sort and plot these distances. We choose ϵ to be the distance where corresponding to the "elbow" or point of maximum curvature on the plot. This is perhaps best demonstrated with an example. When applying this to our problem, we study the distance to the 40^{th} nearest neighbour as we found 40 to be the optimal value for *minpts*. Figure 4a shows the plot of the sorted distances, while Figure 4b shows only the area of most interest to us from the previous plot. Referring in particular to Figure 4b, we can see that the "elbow" of the plot happens when the distance is in the range 1-1.5, therefore we explore values in this range for our value of ϵ.



(a) (b)

Figure 4: The elbow plot we get to estimate the value of ϵ. 4b is the same as 4a but zoomed into the relevant area. We can see here that we want a value of ϵ between 1 and 2.

Using these values we can now try to cluster our data with the DBSCAN algorithm. Applying this to our data with $minpts = 40$ and $\epsilon \in [1, 1.5]$, we obtain 2 clusters, the noise cluster and a core cluster, with the core cluster being comprised 97-99% of the dataset depending on the value of ϵ. This is not the result we would have wished to obtain as it has not split the data as we wished it would, rather only identified some points as noise. We try this method again, however this time subtracting the mean of each time series from the time series. For example for Abbeville County South Carolina we compute the mean of its SAMI over the years 2000-2019 and then subtract this value from the SAMI for 2000, 2001, etc. We do this to eliminate the effect of the values of the SAMI and to focus more on the shape of the time series as it evolves in time. However applying this we obtain the same results as before, a noise cluster and a core cluster. Therefore, we need to find a different method to cluster the data.

4.2.2 Dynamic Time Warping K-Means

K-Means Clustering is a clustering method where the points are clustered based on their distance from a centroid. The algorithm works by first choosing k centroids at random, however still within the distribution of the data. We then go through the dataset and assign each point to the cluster represented by the centroid that is closest to the point. After we have done this to every point in the dataset, we update the position of the centroids by assigning them the average of the points in their cluster. This process is repeated until either no points are reassigned to a new cluster or a maximum number of iterations has been reached. As mentioned at the beginning of §4.2, a problem with using K-Means is that by default the distance metric used is Euclidean Distance. This is not very useful for the our purposes as we wish to cluster the points based on how similar their shape is rather than their distance from each other. However, we can circumvent this issue by using a different similarity metric that takes shape into account. This is where Dynamic Time Warping comes in.

Dynamic Time Warping (DTW) [34] is an algorithm for measuring the similarity of temporal sequences of not necessarily equal length. DTW's ability to match shapes invariant of scale means that it has applications in areas such as automatic speech recognition to recognise recognise voices irrespective of their speed, as well as for pattern recognition with stock market data. This scale invariance is achieved due to the way that DTW matches points in the sequence. While many distance metrics such as Euclidean distance match the i^{th} point in the first sequence with the i^{th} point of the second sequence, this is not possible in the general use case where DTW is used as the sequences in general the same number of elements. Therefore we instead try to match the peaks to the peaks and the troughs to the troughs by creating a one-to-many correspondence between points in the first sequence and points in the second sequence following some rules:

- Every point in the first sequence must correspond with at least one point in the second sequence and vice versa.
- The first element of the first sequence must be matched to the first element of the second sequence, though it may also have other matches.

- The last element of the first sequence must be matched to the last element of the second sequence, though it may also have other matches.
- The mapping of elements is monotonic, that is to say if the i^{th} element of the first sequence matches with the k^{th} element of second sequence and the j^{th} of the first sequence matches with the l^{th} element of second sequence, where $i < j$ then we must have that $k < l$.

The difference between how Euclidean distance and DTW matching points can be seen in Figure 5. The matching that we use for DTW is the matching following these rules that also minimises the cost which we will define next. It is then this cost that we will use as the distance metric between two sequences. DTW cost can be defined with the following algorithm.

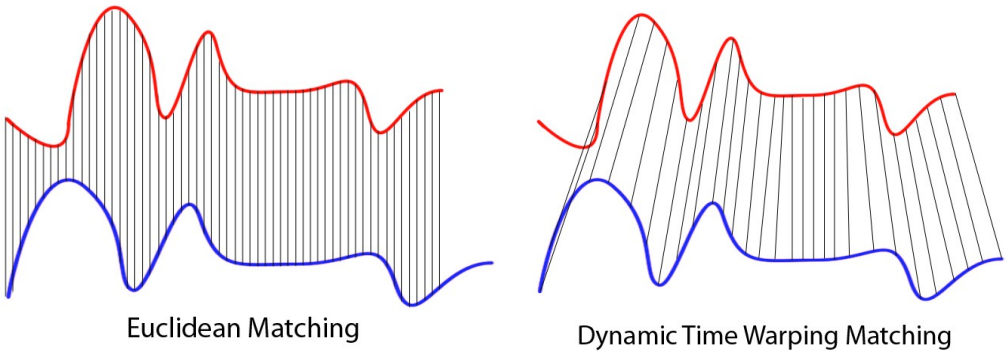


Figure 5: A visual representation of how point matching differs between DTW and Euclidean distance. We see that Euclidean matching only pairs corresponding points, however DTW matching doesn't necessarily require points to be matched to their corresponding point or necessarily to a single point. Image from [35].

Algorithm 1: Dynamic Time Warping

Input : s a sequence of length n, t a sequence of length m

Set DTW to be the $n \times m$ matrix with all entries set to ∞
Set $DTW[0,0] = 0$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** m **do**
    $cost = d(s[i], t[j])$
    $prevs = \min\{DTW[i-1, j],\ DTW[i, j-1],\ DTW[i-1, j-1]\}$
    $DTW[i, j] = cost + prevs$
  end
end
return $DTW[n, m]$

In the algorithm $d(x, y)$ is just any standard metric, here we use Euclidean distance, i.e. $d(x, y) = ||x - y||_2$. Also notice that $DTW[i, j]$ is just the DTW distance between the subsequences $s_1, \ldots, s_i$ and $t_1, \ldots, t_j$.

Now that we have our distance metric between each time series, we can now use it to start clustering the data. Another problem with the k-means algorithm is that it depends on you knowing the number of clusters k that you wish to find. There are ways to overcome this by finding the optimal number of clusters such as Silhouette Analysis [36] and by creating an elbow plot similar to how we did for ϵ in §4.2.1. The problem with these methods, however, is that they require you to perform the clustering for each of the values of k and then to compare certain metrics related to each clustering. Since this requires a large amount of computation, it is not always practical, particularly with large datasets in high dimensions and computationally complex distance metrics such as DTW. Fortunately this shouldn't be a problem for us as we have an idea of the neighbourhood we want k to be in. At the very least we want two clusters, one for stable time series and another for the non-stable time series. However, having more clusters may extract more of the unstable class, for example we may get a cluster where the elements are generally increasing and another that is generally decreasing as well as the stable class. In fact this is what we find.

We first attempt to cluster the data as before without processing it any further. When we do this with k clusters, DTW k-means just divides the data into horizontal bands as can be seen in Figure 6. I believe that this happens due to the way that DTW is defined. As we mentioned previously we still use euclidean distance to measure the distance between points in each sequence, therefore the DTW distance of the two time series is still being dominated by their Euclidean distance as points in 20 space. To solve this as we did in §4.2.1 we subtract the mean of each time series from them. After doing this we obtain better results. A discussed previously we find that using a value of k larger than 2 yields better results as it allows for increasing and decreasing time series to be clustered differently. We find that the best values of k are 3 and 4 for this reason and because they are not too large, unnecessarily dividing the data. Once we have our separate clusters, we need to decide which are the non-stationary clusters. We can do this both visually and by quantifying the average stability of each cluster. Figure 7 shows the result of the clustering process with 3 clusters. We see that Cluster 0 looks like it is the cluster with mostly stable time series, Cluster 1 contains those mostly increasing, and Cluster 2 contains those mostly decreasing. We can confirm this by looking at the average standard deviation in each cluster, which tells us that Cluster 0 has a low average variability, while Clusters 1 and 2 have higher variation. Therefore in this case we combine Clusters 1 and 2 to form our dataset of non-stationary points with which we can build predictive models. We do this for all of the features, *wage*, *estab* and *emp*, in each case getting a non-stationary dataset with about 1000 elements.

Figure 8 visualises where in the contiguous US the counties with non-stationary data are. Figures 8a, 8b, and 8c show the counties where *emp*, *estab* and *wage* respectively are non-stationary. From these plots the overlap of these counties seems to be quite small, which is confirmed by Figure 8d, which shows the counties which are unstable in all three features. In fact only 363 of a total 3109 counties appear in all three features. We also notice that the the majority of these counties in the intersection are situated on the East coast. This could be because of the way that the US historically developed, the East coast is far more densely populated than

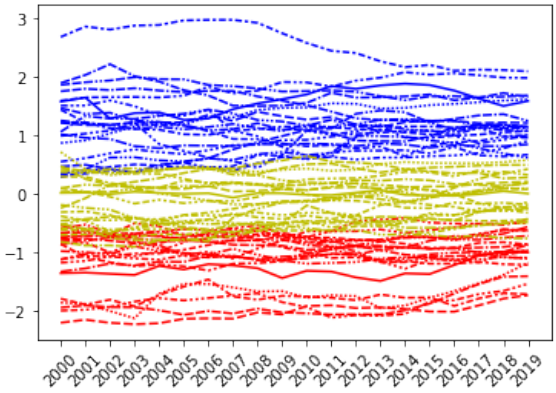


Figure 6: Clustering the data with 3 clusters without first subtracting the mean. Each colour in the plot represents a different cluster. We see in this case k-means with DTW just segments the time series based on their average value.

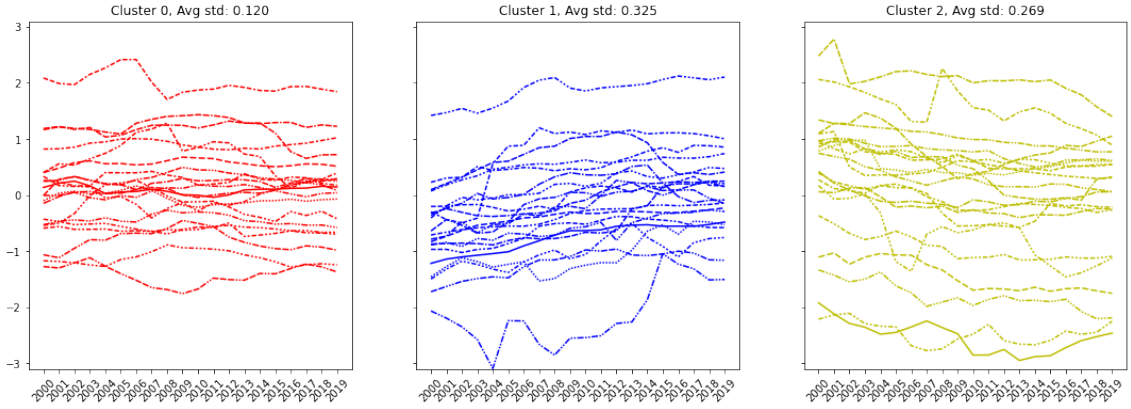


Figure 7: Results when clustering on data where the mean has been subtracted, with $k = 3$. As expected the results are better than those in Figure 6 where we did not subtract the mean. It appears that Cluster 0 contains stationary points while Clusters 2 and 3 contain the non-stationary points, which is confirmed by their average standard deviations.

the rest of the country and therefore better follow the hypotheses of Urban Scaling. Figure 8e, on the other hand shows the counties which are found to be non-stationary in at least one feature. Again there seems to be a preference for the East coast, and notably the entire West coast is missing. In fact there are a total of 7 states with no counties being non-stationary in any feature, these being Connecticut, California, Colorado, Delaware, Arizona, Arkansas, and Alabama. It also could be that some of the missing states could have been removed as anomalous when re-scaling the data in §4.1.

Using this information we can also create two new datasets with which we can build predictive models. While so far we have been looking at the different features separately we can also combine them. This will give us the ability to allow for changes in one feature to affect the outcome of another feature. We will call these new datasets *intersection* and *union*. *Intersection* contains data for all three features, *estab*, *emp*, and *wage*, only for the counties that we found in the intersection before.

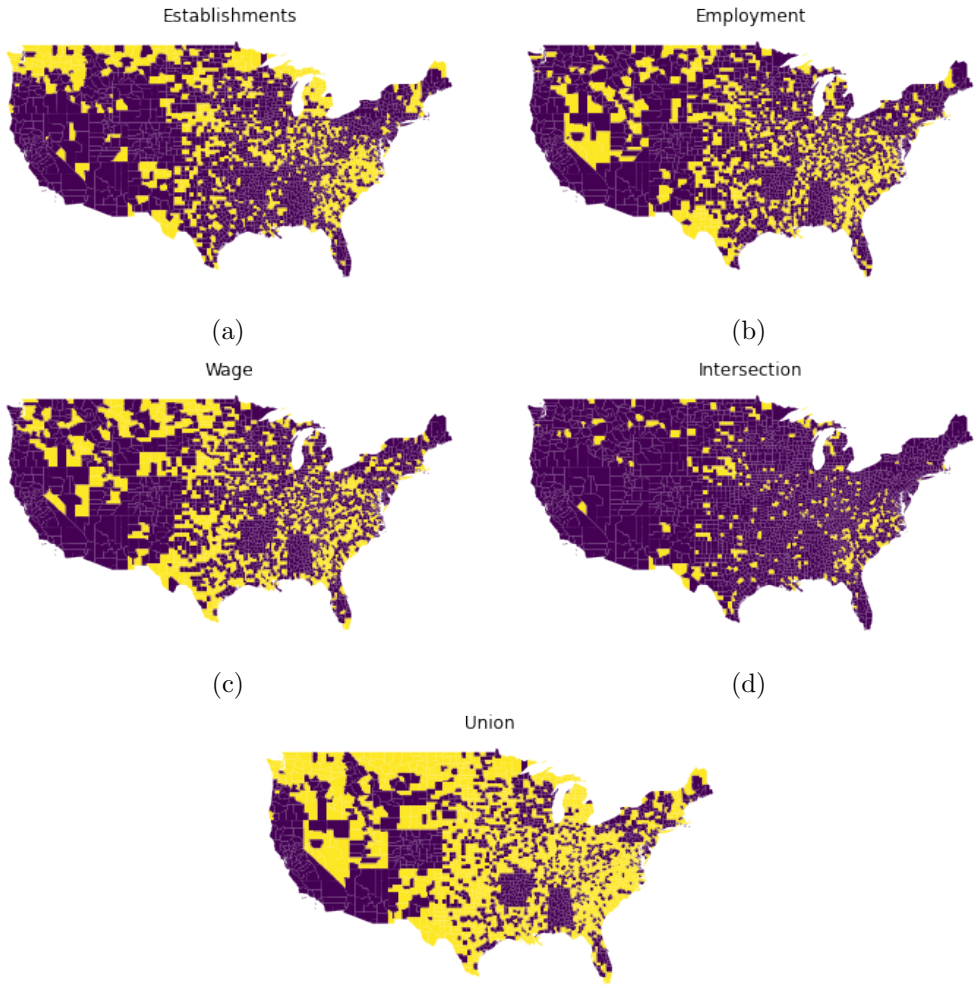


Figure 8: The counties in the Contiguous US with non-stationary time series in each feature. 8a-8c show in yellow the non-stationary counties in their respected feature, and the stationary in purple. 8d shows the intersection of these, that is the counties that are non-stationary in all features, in yellow and the rest in purple. 8e shows the union of 8a-8c in yellow, i.e. the counties that are non-stationary in at least one feature.

Likewise *union* contains data for all three feature but for counties in the union from before.

5 Prediction

Now that we have extracted the non-stationary elements from the dataset, we may now begin to build predictive models on our data. This section is in three parts. We first describe which machine learning methods that we use and the reasons for choosing them, we then discuss the methodologies that we use in our experiments, and finally discuss the results of our experiments.

5.1 Methods

There are a few methods that we use for our models. We will always first use a linear regression as this the simplest model that we can use and if this is good enough there is no need to use more complex methods. Otherwise we will be using some ensemble methods. The idea of ensemble methods is to build an ensemble of multiple predictors and then combining the predictions of these to obtain a final prediction. The methods that we will be using are Random Forests and Gradient Boosting.

5.1.1 Random Forests

A random forest [37] is an example of an algorithm using bagging. Bootstrap aggregating or bagging [38] is a method where subsets of a dataset are drawn at random for training a predictor to increase stability and reduce the chance of overfitting. For a dataset D with N samples we can create a bootstrap dataset B by selecting n elements of X at random with replacement. We call the remainder of the original dataset, i.e. $X \setminus B$, the out-of-bag dataset. The main idea behind random forests is to create M bootstrap datasets and to train a separate regression tree on each of the M bootstrap datasets. Predictions on new testing data can then be made by averaging the prediction from each of the regression trees. This can be summed up in the following algorithm.

Algorithm 2: Random Forest

Input : A dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, the number of trees M and the size of each bootstrap set n

for $m = 1$ **to** M **do**
  Sample $B_m = \{(x_j, y_j)\}_{j=1}^{n}$ randomly from D with replacement
  Train a Regression tree f_m on B_m
end
return $\hat{f} = \frac{1}{M} \sum_{m=1}^{M} f_m$

An advantage of this algorithm is that it is highly parallelisable, since each bootstrap dataset B_m and regression tree f_m are independent of each other. Therefore it can still be quite fast to train even with large values of N, M and n. This method is better than using a single complex base regression tree due to the fact that regression trees have a tendency to overfit to the training data and therefore have a high variance and are highly sensitive to noise. However by building an ensemble of uncorrelated regression trees and taking the average of their predictions we reduce the variance and cancel out the effect of noise. This is why it is important that each f_m is trained on a bootstrapped dataset rather than the main dataset D, the act of using the bootstrapped sets allows the regression trees to be uncorrelated. A disadvantage, however is that since we are partitioning the original training dataset, this algorithm is unsuitable for training on small datasets. Therefore we only use this method if there is a large enough dataset available for training.

5.1.2 Gradient Boosting

Gradient Boosting [39], on the other hand is an algorithm that uses the Boosting [40] principle. The basic principle of boosting is to sequentially build a models that use the error of the previous model to learn. In particular, as the name suggests, Gradient Boosting draws both from boosting and the principle of gradient descent. Where AdaBoost [40] builds a series of predictors, weighting each by their error and summing the weighted predictions to obtain a final prediction, Gradient Boosting instead weights each model by a fixed learning rate $\eta \in (0, 1)$ and at each stage learns the error of the previous model. The purpose of weighting the models by a value less than 1 is because we wish to make small steps at each stage so as to find a globally optimum function and reduce the chance of overfitting. The algorithm for this is as follows.

Algorithm 3: Gradient Boosting

Input : A dataset $D = \{(x_i, y_i)\}_{i=1}^N$, the number of trees M, the learning rate $\eta \in (0, 1)$ and a differentiable loss function L

1 Initialise the model with a constant value $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$

2 **for** $m = 1$ **to** M **do**

3 Compute the pseudo-residuals $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$ for $i = 1, \ldots, N$

4 Fit a regression tree to the r_{im} and create terminal regions R_{jm} for $j = 1, \ldots, J_m$

5 For $j = 1, \ldots, J_m$ compute $\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$

6 Set $f_m(x) = f_{m-1}(x) + \eta \sum_{j=1}^{J_m} \gamma_{jm} \mathbb{I}(x \in R_{jm})$

7 **end**

8 **return** $f_M(x)$

The loss function that is often used for regression problems is the Mean Squared Error (MSE) $L(y_i, \hat{f}(x_i)) = \frac{1}{2}(y_i - \hat{f}(x_i))^2$, with derivative, with respect to $\hat{f}(x_i)$, $-(y_i - \hat{f}(x_i))$. In this case line 1 in the algorithm just becomes computing the mean of the targets y_i, and line 3 becomes $r_{im} = y_i - f_{m-1}(x_i)$. In line 4 J_m is just the number of leaf nodes that the tree we build has and the terminal regions R_{jm} are the sets of elements x_i that terminate leaf node j when passed through the tree.

Since, unlike random forests, gradient boosting uses the whole dataset at each stage rather than partitioning it, therefore is more suitable for learning on small datasets. There is also empirical evidence [14, 15, 41] that gradient boosting models outperform random forests in certain applications. However, since each stage of gradient boosting depends on the outcome of the last stage it cannot be parallelised and therefore may take a long time to train.

5.2 Methodology

We investigate many settings in this problem. As mentioned previously in all settings we first fit a linear regression to as this is the simplest model, and if we don't get good results only then will we apply a random forest and gradient boosting. We will first be training models on data, for each of *estab*, *emp*, and *wage*, from years $t, t+1, \ldots, t+4$ to predict year $t+5$. We then test the performance of the models predicting multiple years in the future, i.e. predicting $t+5$, $t+6$ and $t+5, \ldots, t+9$. We also investigate how the gradients of the time series change. If we view each time series as a function $f(t)$ of time, we have that the definition of the derivative of f with respect to t is

$$f'(t) = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}.$$

However as we only have discrete timesteps of 1 year for t, we must approximate this. The smallest we can let h be without being 0 is 1, so an approximation we could use is $f'(t) = f(t+1) - f(t)$. However this only takes into account "where the function is going" and doesn't consider "where the function has come from". We can make another approximation that does take both into account by using the following manipulation

$$\begin{aligned} f'(t) &= \frac{1}{2}(f'(t) + f'(t)) \\ &= \frac{1}{2}\left(\lim_{h \to 0} \frac{f(t) - f(t-h)}{h} + \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}\right) \\ &= \lim_{h \to 0} \frac{f(t+h) - f(t-h)}{2h}. \end{aligned}$$

Therefore we can better approximate $f'(t)$ as $f'(t) = \frac{1}{2}(f(t+1) - f(t-1))$. By the same process we can approximate the second derivative as $f''(t) = \frac{1}{2}(f'(t+1) - f'(t-1)) = \frac{1}{4}(f(t+2) - f(t-2))$. For both the first and second derivative we perform the same experiments as with the base data, however we also try train on the base data for years $t, t+1, \ldots, t+4$ to predict the first and second derivatives of the subsequent years. We also apply all of the same experiments on the combined datasets *union* and *intersection*, however since *intersection* contains relatively few elements we don't apply a random forest to it, only linear regression and gradient boosting.

For all of these experiments we train a model on data from the first 5 years available, 2000-2004 for the base data, 2001-2005 for the first derivative and 2002-2006 for the second, and apply them to different time periods. This is to see how well the models generalise. To evaluate the models, we use the built in sklearn score function which uses the coefficient of determination, denoted by R^2. This is a measure of how well a regression model replicates the actual observations. For a model that perfectly predicts the observations we get an R^2 of 1, an R^2 of 0 for models that predict the observation mean for all instances, and an R^2 less than 0 for models that perform worse than predicting the mean. It is computed in the following way.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{f}(x_i))^2}{\sum_i (y_i - \overline{y})^2}$$

Where $\overline{y}$ is the observation mean.

5.3 Results

When predicting a single year we find that with a linear regression model for *emp*, *estab* and *wage* we obtain an excellent training score of 0.97-0.99. Looking at the coefficients of the models we see that they are predicting a scaling of the last year of 1.1-1.2 and subtracting as small scaling of previous years. We obtain a similar performance when applying the model to different time periods. For *emp* this is 0.99 across the board, however for *estab* and *wage* this is slightly lower with R^2 ranging between 0.95 and 0.98 which are still excellent results.

As we add more years to the prediction, i.e. predicting 2005 and 2006 and 2005-9, we still see similar results, however, as you would expect, the further into the future you try to predict the worse the models become. For the two year prediction of *emp* we achieve a training score of 0.96 and 0.92 for the 5 year prediction. These translate to test scores of 0.95-0.95 and 0.91-0.93 respectively. Similarly for *estab* we get a train score of 0.98 and test scores 0.97-0.99 for the two year prediction and a train score of 0.96 and test scores 0.93-0.96 for the ten year prediction. As before, the scores for *wage* were slightly lower with test scores 0.94-0.96 and 0.88-0.91 for the two and ten year predictions respectively.

We again, get very similar results when we combine the features in *union* and *intersection*. For *intersection* we obtain test score in the range 0.94-0.98 for the single prediction, opposed to a slightly higher 0.97-0.99 for *union*. This pattern continues for the two and ten year predictions. This slightly better result for *union* could be due to the fact that there are more elements in it to fit a linear model to. However, this could also be that every instance of *intersection* is non-stationary in all three features and is therefore harder to fin a linear model to. Something we notice in the models for *union* and *intersection* is that the coefficients aren't just copies of the cases with single feature, indicating that each of the indicators does contain some information that can be used to help predict the others. Since we have obtained such high scores we can see that for predicting the base indicators, a linear model is the most suitable.

When it comes to predicting the derivatives we get another story altogether. We find that the performance of linear models in this case is far worse than in the case of predicting the base data. The best R^2 achieved in training was around 0.5 with similar the test scores being similar or worse. Therefore we use gradient boosting and random forests. We find that for in this task, contrary to the what current research suggests [14, 15, 41], the random forests performs far better in training than gradient boosting. For example for predicting a single year of the first derivative for *wage*, gradient boosting achieved a training R^2 of 0.77, while a random forest achieved 0.91. While this is the case for training, we observe that the test results are very poor. We obtain R^2 at best 0.4-0.5 for single year predictors, however this would not be for all test periods. It seems that the models perform best on very early and later time periods, specifically they perform worst on periods containing the 2008 financial crash and the few years after. We also obtain very similar results on the combined features datasets *union* and *intersection*.

The fact that we get good training scores and bad testing scores could mean one of two things. The first is that our models are overfitting to the training data and

should be tuned via cross validation. However both random forests and gradient boosting are in theory quite resistant to overfitting [41–43]. The second possibility is that while it is possible to learn the derivatives on a period, these models doesn't generalise well to other periods.

6 Conclusion

The ways that cities and their statistics grow and change over time has become a popular area of research in recent years. One of the most popular phenomena in this area has become that of observing how urban indicators follow scaling laws and power laws with their population. These scaling and power laws have allowed for new rescaled statistics to be created that allow for better comparison of cities with vastly different populations. There has recently been work to model the change of these rescaled statistics via statistical methods such as Maximum Entropy models. This work was able to establish a robust model, particularly for cities with populations grater than 10^4.

We find that the scaling laws formalised for cities work with US counties as well. This is not something that is necessarily expected as counties do not follow the same hypotheses of a city, for example they are not as densely populated and in some cases have very small populations. Using these scalings we are able to rescale the data, obtaining a distribution close to that of a Gaussian with mean of 0. Analysis of the coefficients obtained from the scaling show that they reflect certain aspects the economic setting. We see that the rates of change of coefficients increase dramatically around 2007-2009 and beginning to stabilise in the early 2010s, corresponding to the financial crash and its recovery. By applying the rescaling we "hide" this effect as we centre the data for each year about the average behaviour for that year, therefore for years where all of the data shows abnormal behaviour we treat the behaviour as normal. It would be interesting for future work in this area to perhaps define and then use a new way of scaling the data that doesn't have this effect. In doing this it may be possible predict when economic events such as an economic crash may happen.

After applying the rescaling the majority of the instances in the dataset were seen to be stationary, taking very similar values for each year. This provides an issue for machine learning methods as it is likely that the models will be skewed by the stationary points and learn to predict the same value for each year. This would lead to poor performance when predicting the non-stationary points. We find that this problem can be solved by extracting the non-stationary elements using clustering methods with Dynamic Time Warping as our distance metric.

For forecasting the rescaled indicators we find that linear models achieve excellent results and generalise just as well to periods that they are not fit to. However, when it comes to fitting linear models to approximations of the derivative we see the opposite. When using more complex methods, random forest and gradient boosting, we are able to attain excellent results on our training set, however when we apply these models to different time periods we achieve very low scores. This could be either a problem of overfitting or there not being a general pattern in the derivatives of the time series. Further work could be done in this area to confirm either of these

two hypotheses, by attempting hyper-parameter tuning for the machine learning methods via cross validation, or the exploration of further machine learning methods that may be better suited to finding the patterns in this data.

References

[1] L. G. Alves, H. V. Ribeiro, and F. A. Rodrigues, "Crime prediction through urban metrics and statistical learning," *Physica A: Statistical Mechanics and its Applications*, vol. 505, pp. 435–443, 9 2018.

[2] T. P. Davies, H. M. Fry, A. G. Wilson, and S. R. Bishop, "A mathematical model of the London riots and their policing," *Scientific Reports*, vol. 3, no. 1, pp. 1–9, 2 2013. [Online]. Available: https://www.nature.com/articles/srep01303

[3] B. Monechi, M. Ibáñez-Berganza, and V. Loreto, "Hamiltonian modelling of macro-economic urban dynamics," *Royal Society Open Science*, vol. 7, no. 9, p. 200667, 9 2020. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsos.200667

[4] H. Youn, L. M. Bettencourt, J. Lobo, D. Strumsky, H. Samaniego, and G. B. West, "Scaling and universality in urban economic diversification," *Journal of the Royal Society Interface*, vol. 13, no. 114, 2016.

[5] L. M. Bettencourt, "The origins of scaling in cities," *Science*, vol. 340, no. 6139, pp. 1438–1441, 2013.

[6] United Nations Department of Economic and Social Affairs Population Division, "68% of the world population projected to live in urban areas by 2050, says UN — UN DESA — United Nations Department of Economic and Social Affairs," pp. 2–5, 2018. [Online]. Available: https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

[7] L. M. Bettencourt, J. Lobo, and D. Strumsky, "Invention in the city: Increasing returns to patenting as a scaling function of metropolitan size," *Research Policy*, vol. 36, no. 1, pp. 107–120, 2 2007. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0048733306001661

[8] L. M. A. Bettencourt, J. Lobo, and H. Youn, "The hypothesis of urban scaling: formalization, implications and challenges," pp. 1–37, 1 2013. [Online]. Available: http://arxiv.org/abs/1301.5919

[9] L. M. A. Bettencourt, J. Lobo, D. Strumsky, and G. B. West, "Urban Scaling and Its Deviations: Revealing the Structure of Wealth, Innovation and Crime across Cities," *PLoS ONE*, vol. 5, no. 11, p. e13541, 11 2010. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0013541

[10] A. Carbone, P. Murialdo, A. Pieroni, and C. Toxqui-Quitl, "Atlas of Urban Scaling Laws," 9 2021. [Online]. Available: https://arxiv.org/abs/2109.11050v1http://arxiv.org/abs/2109.11050

[11] L. M. A. Bettencourt, V. C. Yang, J. Lobo, C. P. Kempes, D. Rybski, and M. J. Hamilton, "The interpretation of urban scaling analysis in time," *Journal of the Royal Society Interface*, vol. 17, no. 163, 2 2020. [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2019.0846

[12] C. W. Gardiner, *Handbook of Stochastic Methods: for Physics, Chemistry and the Natural Sciences*, 3rd ed. Springer, 2004.

[13] H. R. Varian, "Big data: New tricks for econometrics," *Journal of Economic Perspectives*, vol. 28, no. 2, pp. 3–28, 5 2014. [Online]. Available: http://dx.doi.org/10.1257/jep.28.2.3https://pubs.aeaweb.org/doi/10.1257/jep.28.2.3

[14] S. M. Piryonesi and T. E. El-Diraby, "Data Analytics in Asset Management: Cost-Effective Prediction of the Pavement Condition Index," *Journal of Infrastructure Systems*, vol. 26, no. 1, p. 04019036, 12 2019. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29IS.1943-555X.0000512

[15] S. Madeh Piryonesi and T. E. El-Diraby, "Using Machine Learning to Examine Impact of Type of Performance Indicator on Flexible Pavement Deterioration Modeling," *Journal of Infrastructure Systems*, vol. 27, no. 2, p. 04021005, 6 2021. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%29IS.1943-555X.0000602

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 1 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[17] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, "Tslearn, A Machine Learning Toolkit for Time Series Data," *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-091.html

[18] OECD, "The OECD Metropolitan Areas Database," 2021. [Online]. Available: http://stats.oecd.org/Index.aspx?DataSetCode=CITIES

[19] U.S. Bureau of Labor Statistics, "QCEW Data Files : U.S. Bureau of Labor Statistics," 2020. [Online]. Available: https://www.bls.gov/cew/downloadable-data-files.htm

[20] "Methodology for the Intercensal Population and Housing Unit Estimates." [Online]. Available: http://www.census.gov/popest/research/modified.html.

[21] "Population Base Births Deaths Migration Population Estimate," 2010. [Online]. Available: http://www.census.gov/programs-surveys/popest/guidance-

[22] "Insee - Institut national de la statistique et des études économiques." [Online]. Available: https://www.insee.fr/fr/accueil

[23] B. Monechi, "Dataset for "Hamiltonian Modelling of Macro-Economic Urban Dynamics"," 2020. [Online]. Available: https://doi.org/10.6084/m9.figshare.12196152.v1

[24] A. Murphy, "An Analysis of the Financial Crisis of 2008: Causes and Solutions," *SSRN Electronic Journal*, 11 2008. [Online]. Available: https://papers.ssrn.com/abstract=1295344

[25] C. M. Reinhart and K. S. Rogoff, "Recovery from Financial Crises: Evidence from 100 Episodes," *American Economic Review*, vol. 104, no. 5, pp. 50–55, 5 2014. [Online]. Available: https://pubs.aeaweb.org/doi/10.1257/aer.104.5.50

[26] J. Foo and D. Witkowska, "A Comparison of Global Financial Market Recovery after the 2008 Global Financial Crisis," *Folia Oeconomica Stetinensia*, vol. 17, no. 1, pp. 109–128, 6 2017. [Online]. Available: https://www.sciendo.com/article/10.1515/foli-2017-0009

[27] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," 2 2018. [Online]. Available: http://arxiv.org/abs/1802.06971

[28] C. Jiashun, "A new trajectory clustering algorithm based on TRACLUS," in *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*. IEEE, 12 2012, pp. 783–787. [Online]. Available: http://ieeexplore.ieee.org/document/6526048/

[29] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, "A review of moving object trajectory clustering algorithms," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 1 2017. [Online]. Available: http://link.springer.com/10.1007/s10462-016-9477-7

[30] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," 1996. [Online]. Available: www.aaai.org

[31] "DBSCAN Parameter Estimation Using Python — by Tara Mullin — Medium." [Online]. Available: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd

[32] J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications," *Data Mining and Knowledge Discovery 1998 2:2*, vol. 2, no. 2, pp. 169–194, 1998. [Online]. Available: https://link.springer.com/article/10.1023/A:1009745219419

[33] Z. Yan, J. Yang, Y. Su, a. , H. Gwalani, and B. Gwalani, "Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra," *IOP Conference Series: Earth and*

Environmental Science, vol. 31, no. 1, p. 012012, 1 2016. [Online]. Available: https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/meta

[34] R. E. Bellman, "Adaptive Control Processes," *Adaptive Control Processes*, 12 1961.

[35] "Dynamic Time Warping. Explanation and Code Implementation — by Jeremy Zhang — Towards Data Science." [Online]. Available: https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd

[36] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. C, pp. 53–65, 11 1987.

[37] T. K. Ho, "Random decision forests," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 1, pp. 278–282, 1995.

[38] L. Breiman, "Bagging Predictors," 1994.

[39] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Online]. Available: http://www.jstor.org/stable/2699986

[40] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Society*, vol. 14, no. 5, pp. 771–780, 1999.

[41] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009. [Online]. Available: http://link.springer.com/10.1007/978-0-387-84858-7

[42] A. Vezhnevets and O. Barinova, "LNAI 4701 - Avoiding Boosting Overfitting by Removing Confusing Samples," *LNAI*, vol. 4701, pp. 430–441, 2007.

[43] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 6 2016. [Online]. Available: https://link.springer.com/article/10.1007/s11749-016-0481-7