University of Copenhagen
Department of Economics

# Can Google searches & Machine Learning predict the Danish unemployment rate

**Social Data Science, Summer School 2018**

Submitted 1 September 2018

# Contents

Exam number 89 has contributed to the following sections, cf. the table of contents:

- 1, 2, 3.4, 4.4, 5.1, 6, 7.3, 8

Exam number 104 has contributed to the following sections, cf. the table of contents:

- 1, 3.1, 4.1, 4.5, 5.2, 7, 8

Exam number 183 has contributed to the following sections, cf. the table of contents:

- 1, 3.2, 4.2, 4.6, 5.3, 7.1, 8

Exam number 203 has contributed to the following sections, cf. the table of contents:

- 1, 3.3, 4.3, 5, 5.4, 7.2, 8

# 1 Introduction / Motivation

The internet has become an increasingly important source of information in the 21st century. According to the European Commission, Denmark is the frontrunner of digitilisation in Europe[1]. Over the past three years, Google was Denmark's number one search engine, with a market share of 95 pct[2]. Indeed, Google facilitates people's access to information and can contribute to their decision making, but can people's searching behaviour describe their incentives?

This paper has two aims. First, we try to predict the change in Danish the unemployment rate one month ahead based on volume of job-related Google searches. Our predictions are only valid for stable periods in the economy, i.e. we do not claim or attempt to be able predict shocks such as the GFC. We perform multivariate time series regressions in first differences to make one-step predictions based on supervised machine learning. Our hypothesis is that the number of job-related searches is correlated with the number of people who are unemployed, soon-to-be-unemployed or afraid of being laid off soon e.g. during recessions. This link has been proven before in other countries, but not in the Danish context. Our second aim is to test if we can improve the forecasting ability by using machine learning instead of econometrics.

We find that the best variant of our models including Google searches performs up to 19.3 pct. better than the standard econometric model, i.e. autoregressive model (AR(1)). And that the Random Forest algorithm is a better predictor of the unemployment rate than the Lasso and Ridge method and the Decision Tree algorithm.

In the following section we review a few articles that have been investigating the same correlation between Google searches and the unemployment rate. Next we present the data for our analysis, i.e. data for unemployment and data for Google search queries. Then we present the theory of Machine Learning which includes the Lasso and the Ridge, and two non-linear algorithms, Decision Tree and Random Forest. Afterwards we present our empirical analysis and our results, which we compare to a standard autoregressive model, i.e. AR(1). Finally, we discuss or results, the weaknesses of our analysis and how we could extend it and compare the results with the existing literature.

[1] http://ec.europa.eu/information_society/newsroom/image/document/2018-20/dk-desi_2018-country-profile_eng_B43FFE87-A06F-13B2-F83FA1414BC85328_52220.pdf

[2] http://gs.statcounter.com/search-engine-market-share/all/denmark/monthly-201507-201807

# 2 Literature review

This section reviews a few articles which investigates the same correlation between Google searches and the unemployment rate.

In their paper "Can Google econometrics predict unemployment? Evidence from Spain" from 2018, Marcos Gonzales-Fernandez and Carmen Gonzales-Velasco find a high correlation between Google searches and the unemployment rate in Spain with monthly data from January 2004 to November 2017. In their analysis they compare a baseline model, a simple AR(1) model, with the model with internet search queries about unemployment, and then with a model with lagged search queries about unemployment. Their results show that the model where search queries are included without any lags is in fact a better model for prediction than their baseline of a simple AR(1), since this model will produce the smallest root mean squared errors in the prediction of unemployment, and is a 18.9 pct. better predictor than AR(1).

Francesco D'Amuri and Juri Marcucci also find a high predicting power on unemployment in the US from Google searches in their article: "The predictive power of Google searches in forecasting US unemployment" from 2017. They use both a long and short sample size for their predictions, the short sample going from 2004.2 to 2014.2 and long sample going from 1997.7 to 2014.2. Overall, the short sample size seems to be a better predictor compared to the baseline of a basic AR(1) model than the long sample. Their results show that the models using Google search queries achieves the best performance when it comes to root mean squared forecast errors (RMSFEs) with an advantage over the baseline, and it actually increases with the forecast horizon (18 pct. improvement one time step ahead).

In the literature we find a difference in what the different researchers chooses as search keywords on Google when predicting the unemployment rate. Gonzalez-Fernandez & Gonzalez-Velasco (2018) chooses only the word "desempleo", the Spanish word for unemployment, since they assume that this word will summarize the search queries about the situation of loosing a job. Next, other words like payment received when unemployed is in Spanish "subsidio por desempleo", so these queries will already be gathered with the word "desempleo".

D'Amuri & Marcucci (2017) chooses only the word "jobs" for their queries in their article, since they find this word to be the most popular among different job-search-related keywords in the US. Next, they argue that it is the word that is used most widely across job seekers and

thereby is less sensitive to the presence of demand and supply shocks specific to subgroups of workers, which could then bias the value of the Google search index. Finally, the word "jobs" is not only being used by unemployed people but also employed people, and it thereby captures the overall job-search activity.

Like many other articles within the literature, both Gonzalez-Fernandez & Gonzalez-Velasco (2018) and D'Amuri & Marcucci (2017) gather their internet search data from Google Trends which provides the Google search index (GI). An explanation of this index will follow in the data section below.

# 3 Data

In this section we present the data for both the Danish unemployment rate and the data used for search queries on Google relating to unemployment.

## 3.1 Danish data for unemployment

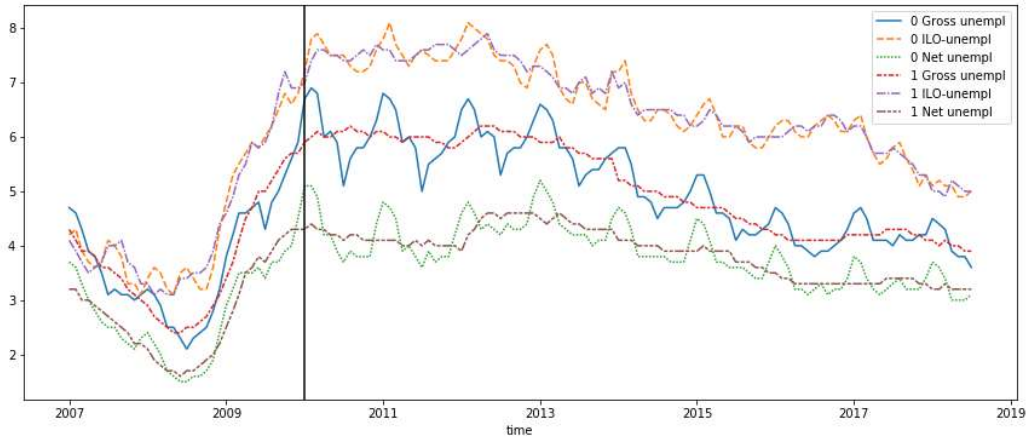
There are three different measures for unemployment in Denmark. Gross unemployment, net unemployment and the ILO unemployment rate (AKU arbejdsløshedsmålet - Arbejdsmarked-sundersøgelsen). The gross and net rates are register based measures, while the ILO rate is a survey based measure of unemployment. The three measures are published on a monthly basis, and they are generally defined as the unemployed share of the total labour force. Figure 1 plots the three measures both seasonal and non-seasonal on a monthly basis for the period 2007/01 to 2018/06 collected from Statistics Denmark. As Figure 1 shows, the level in the three measures seems to differ, but the development tends to correlate.

The register based measures include people aged 16-64, exclude students and pensioners. Furthermore persons who are part-time unemployed are aggregated and converted into full-time unemployed. The gross unemployment includes all whom receive unemployment benefits from the state or an unemployment insurance fund. The net measure is the gross measure net of those in 'activation' (mandatory work activity to keep unemployment benefits, in Danish: 'aktivering').

Finally the ILO unemployment rate follows the International Labour Organization's measurement standards for unemployment and their definition of it: "people completely out of a

job and are available and are or has been job seeking.".

Figure 1: Gross-, net- and ILO unemployed, 2007/01 - 2018/06, pct.



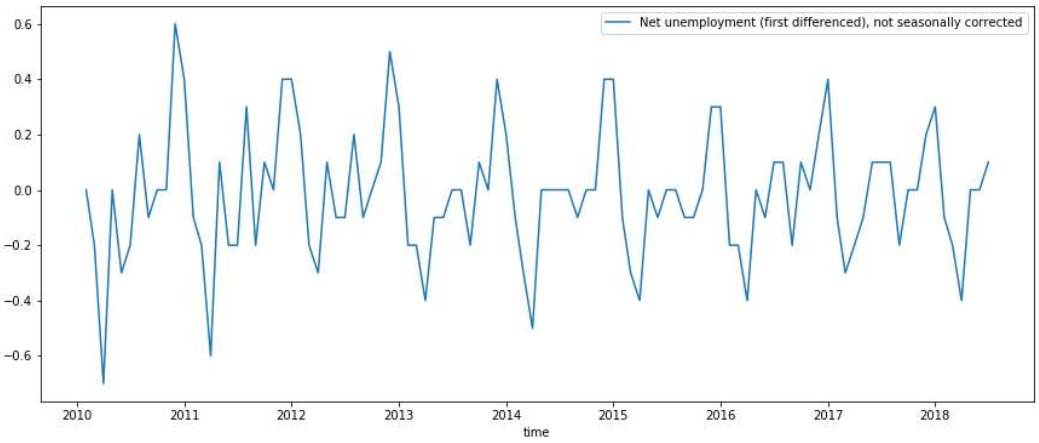

Source: Statistics Denmark

We use the net unemployment rate in our analysis and prediction. Our reason being, that we expect that people who are "activated", and thereby included in the gross measure, would not spend as much time doing job related searches on Google. Next, we choose to collect non-seasonal adjusted data, even though unemployment data is clearly affected by seasonality (it is high in winter and low in summer time). This is due to the fact that we need to be consistent in our data selection and Google Trends are not seasonally adjusted either.

In the literature, researchers are divided when deciding whether to use seasonal or non-seasonal adjusted data. D'Amuri & Marcucci (2017) chooses to work with weakly seasonally adjusted unemployment data whereas Gonzalez-Fernandez & Gonzalez-Velasco (2018) uses non-seasonally adjusted data. D'Amuri & Marcucci (2017) suggest that it should be adjusted by using the X-13 ARIMA-SEATS filtering. However, we considered that it is better to work with raw data and decided not to adjust for seasonality for several reasons. First, the seasonality adjustment is done for showing the directions of the trend. By applying it, the output of the forecasting model would also be the rolling average. It is not convenient as we are interested in predicting the real unemployment rate. Second, it is recommended that both GI and unemployment rate should be adjusted in the same time. However, we consider that it would bias the results of the analysis by changing the correlation.

In order to apply Machine Learning in this project we need a stationary time series[3]. Therefore we choose to drop the first part of our sample, i.e. the data from 2007 to 2010 where we see a big level shift. This we show with the separation line in Figure 1. We are aware that this will exclude the ability for our model to predict large shocks to unemployment, which is unfortunate. Next we take the first difference in order to be absolutely sure that the unemployment rate is in fact stationary. So we choose our final data sample for net unemployment to be the period from 2010/01 to 2018/06. The net unemployment in first difference is shown in Figure 2 below.

Figure 2: Net unemployed, first difference, non-seasonal adjusted, 2010/01 - 2018/06, pct.




Source: Statistics Denmark

## 3.2 The Google Index

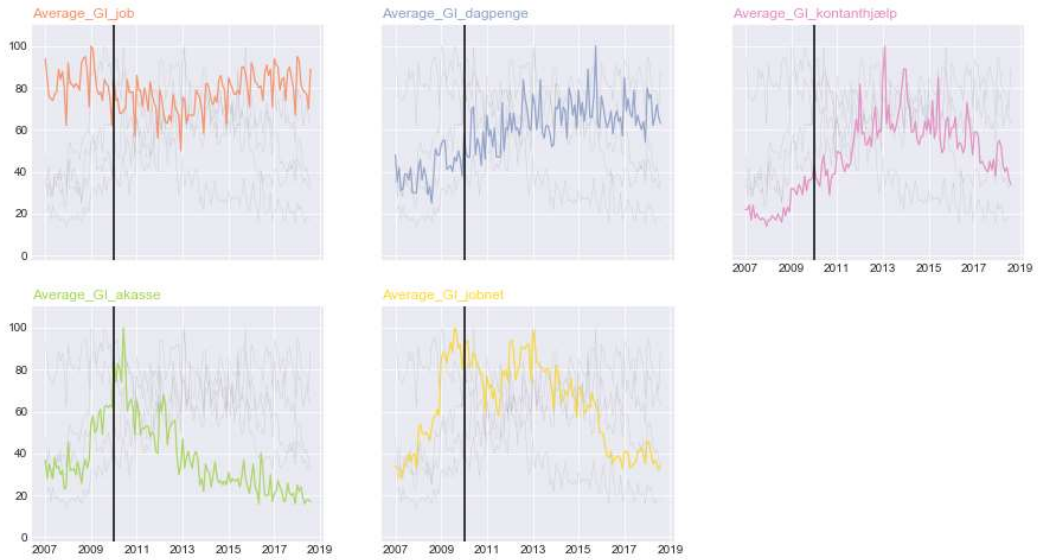
The Google Index (GI) gives the user the possibility to select different search keywords in a given geographical area in a given time frame. The GI provides the indexed numbers of searches, where index 100 is the month with maximum number of searches in the sample. Thus, the GI index represents the likelihood of a random user doing a Google search for the chosen keyword in the specific area and specific time. Finally, it should be mentioned that the index is calculated based on a sample of IPs that changes with time and with the IP address. Therefore, indices can vary according to the specific day, time, and IP used when downloading

[3]https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424

the data (D'Amuri and Marcucci (2017)).

We choose to scrape GI data from Google Trends for the following keywords: "Job", "Kontanthjælp", "A-kasse", "Jobnet" and "Dagpenge". These words are chosen, since they seem to be correlated with the unemployment rate and are assumed to capture most of the search queries related to unemployment in Denmark. This list could of course be extended and or reduced to more or less keywords. A glance below at Figure 3 reveals that some keywords seem to correlate more with the unemployment rate than others. For instance, the word *a-kasse* spikes along with the unemployment increase that came along with the GFC. In contrast, the word *job* does not seem to follow the movements of the unemployment rate of all. We have chosen to keep all five words. Our machine learning models should be able to, at least for the most part, overlook those keywords that do not correlate with the unemployment, because these keywords do not help the machine in prediction.

Figure 3: Google Index for different keywords, 2007/01 - 2018/06




Source: Google Trends

## 3.3 Data collection

The data of interest is available on www.statistikbanken.dk in Statistics Denmark openly provided statistics "AKU125: Labour Force Status" and "AUS07: Unemployed persons". We pull

the data as csv-files from DST's API, using a link constructed through DST's API link console [4]. The constructed link filters the data of interest:

- unemployment in pct. of the total labour force.
- for all months 2007/01 to 2018/06.
- both seasonally corrected and non-seasonally corrected unemployment rates.
- for both ILO-unemployment rate (AKU-arbejdsløshed), Gross Unemployment, Net Unemployment.

The data from Google Trends consists of time series for each search query keyword. The time interval is set to 2007/01-2018/06 and we only look at Google Trends for searches in Denmark. We enter the keywords on Google Trends [5] and scrape the associated time series data one by one. The scrape is then performed using GET requests, which returns responses in JSON-format. From the JSON data we extract the data for each keyword and wrangle them into a single combined dataframe.

Finally we drop the GI data from 2007 to 2010 in order to obtain the consistency in our time series, so that our final GI keyword data runs from 2010/01 to 2018/06. This is shown by the black lines in Figure 3.

# 4 Theory

In this section we present the theory of Machine Learning we apply in order to predict the change in the Danish unemployment rate using the above mentioned search keywords in Google.

## 4.1 Regularization

A common problem when using machine learning is overfitting of the model. This implies that model includes features that actually fit the training data too well. The result of overfitting is reduced prediction power, since the model, even though suitable for describing the training data, cannot explain the out-of-sample data, e.g. the test data. There exist multiple machine

[4] https://api.statbank.dk/console
[5] https://trends.google.com/

leaning methods that account for overfitting. These are so-called regularization methods, and in this paper we will use two of these, the Least Absolute Shrinkage and Selection Operator (Lasso) and the Ridge regression.

The purpose of the regularization methods is to alter the optimization problem in a way, that favours less complex models, i.e. models that do not contain either a large amount of parameters or parameters of large size. Formally we can express the problem as follows:

$$arg \min_{\beta} E[(y_0 - \hat{f}(x_0))^2] + \lambda R(\beta) \tag{4.1}$$

The first term of this problem is rather similar to the optimization problem faced when doing e.g. OLS - we want to minimize the sum of squared errors. The second term is the penalty term, which consists of a hyperparameter $\lambda$ and a function $R$ of $\beta$, that determines exactly how the overparameterization should be penalized. The penalty is what distinguishes the Lasso method from the Ridge. In the Lasso method, the function $R(\beta)$ is specified as:

$$\sum_{j=1}^{p} |\beta_j| \tag{4.2}$$

This implies, that the optimization problem is penalized with the product between $\lambda$ and the absolute sum of all parameters in the model, resulting in a reduced absolute sum in optimum. The Lasso method is effective in preventing irrelevant variables to have an effect, i.e. it counteracts increasing number of features.

The Ridge includes a slightly different function $R(\beta)$:

$$\sum_{j=1}^{p} \beta_j^2 \tag{4.3}$$

The sum of squared parameters ensures, that relatively large parameters will be penalized to a greater extent. Thus, the Ridge cause a reduction of the coefficients' size.

## 4.2 Decision Tree Learning

The first non-linear algorithm we apply is the Decision Tree. The Decision Tree regression builds a regression based on a tree structure, which splits the data into two, and again splits

each subset into a new split and so forth, until the maximum level of depth is reached. In the case of continuous input a split is a division of a feature by imposing a threshold on the feature values. The command goes: Put all data in which the values of feature X are lower in one branch and the rest of the feature X values into another branch. Repeat the process for each of the two newly created datasets, with the same or another feature, but with another threshold.

Each split is made according to an algorithm that maximizes the information gain or correspondingly minimizes the impurity of the datasets. Impurity at a given node is measured as the within-node variance i.e. the mean squared error (MSE). MSEs are often converted to root mean squared error (RMSE), which is a better measure for comparison of the different methods and algorithms within Machine Learning.

At the end of all branches are leaf nodes. Leaves are cannot be split anymore, and thus contains the smallest allowed size of a data size.

It is not necessary to transform our features into linear data when working with the decision tree for a regression. However, we need a metric that is suitable for continuous variables. Therefore, we apply the following measure for MSE depending on the number of nodes, $t$:

$$MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} (y^i - \hat{y}_t)^2 \tag{4.4}$$

$N_t$ is the number of training samples at node $t$. $D_t$ is the training subset at node $t$. $y^i$ is the true target value and $\hat{y}_t$ is the predicted target value. There is a few limitations when using a decision tree as a regressor: The algorithm does not capture the continuity and the differentiability of the desired prediction. Next, like with the standard decision tree, we need to be very careful when choosing the value for the depth of the three and not to overfit or underfit our model (Raschka & Mirjalili (2017)).

## 4.3 Random forest

The second non-linear algorithm we apply is Random forests, which are simply ensembles of decision trees. The *greediness* of Decision Trees if not pruned, can result in high variance and overfitting, which can be mitigated by bootstrapping. Bootstrapping in this context means growing many decision trees and then smooth out their differences. A forest of identical trees is avoided by resampling and training each tree on a random subset of features.

## 4.4 Hyperparameters for Decision Trees

Decision Trees algorithms have many hyperparameters and Random Forests even more. The most important ones are: - Max. depth: limits the levels of splits. - Max. features: limits the number of features considered before splitting - within a tree. - Min. sample leaf: Limits the minimum size of a leaf, i.e. the minimum number of samples. - Min. samples split: Limits the minimum percentage of samples that must be considered in order to split a node. - no, of estimators: Limits the number of trees. NB: Only relevant for random forests. - bootstrap: A boolean, which allows or denies resampling in the observation samples used for each tree in a Random Forest.

Tuning hyperparameters presents a computational challenge when optimizing many hyperparameters, because optimization in many dimensions create a large numbers of candidate combinations of hyperparameter values. The challenge is further exacerbated if some hyperparameters can take many different values. There are two methods to tune multiple hyperparameters. Grid Search, which looks through each possible combination of candidates. The computationally less demanding RandomizedSearch does in contrast to GridSearch not try all parameter values - instead it picks and compares only some candidates, each created by sampling from some specified hyperparameter distributions[6].

## 4.5 Cross-validating time series

To measure the predictive power of a model, the model should be tested not only on in-sample data but also on out-of-sample data, i.e. cross-validation. In machine learning, popular cross-validation methods are train-test-splits and K-fold. In time series however, data is not independent due to serial correlation, and the aforementioned methods cannot readily be applied.

The train-test-split technique can be altered to time series, by making a split that keeps the order of observations. This is also known as fixed-origin evaluation (Tashman (2000)). For our data, that means splitting so that the first months are used for model development (i.e. training and validation) and the remaining, most recent, months are used only for testing. A thorough discussion of this and similar techniques can be found in the article by Tashman (2000).

A widely used, but simple advancement of the fixed-origin evaluation is the rolling-origin-

[6]Source: SCI-KIT documentation

update evaluation (Bergmeir & Benitez (2012)). The data is split multiple times at different points. Although the point of the splits varies and the size of the training data thereby varies, the size of the test set always remain the same. That means that some of the most recent data is unused in some of the splits and it therefore also requires a sufficiently long dataset.

The K-fold method is not usable for our data because of the serially correlated nature of time series and the non-stationarity of the unemployment rate. K-folds can however be used for purely autoregressive time series as it has been shown by Bergmeir *et al.* (2018) but since our proposed prediction model is not purely autoregressive this caveat is not relevant.

# 5 Empirical analysis and results

This section explains how we attempt to predict the change in the unemployment rate using machine learning, and how we measure the performance of our models. Since we wish to predict the change of the unemployment rate in the current period, $\Delta y_t$, we specify a model that include searches for our keywords in the previous period, $X_{t-1}$:

$$\Delta y_t = k_0 + \beta_i X_{t-1} \tag{5.1}$$

Thus $X_{t-1}$ is a vector containing the lagged GI values for the five keywords of interest and $k_0$ is a constant. In the following we estimate this model using two linear and two non-linear algorithms: Lasso, Ridge, Decision Tree Regression and Random Forest Regression respectively. We optimize the hyperparameters of all models. Finally we compare the four models with a simple persistence model, i.e. a purely autoregressive model of order 1, AR(1), estimated by OLS.

## 5.1 Splitting the data: training and test data

As mentioned above, a time series dataset should not be split randomly, but must be split in continuous and coherent blocks as per the fixed-origin evaluation technique. Thus, we divide the entire dataset (2010/01 - 2018/06) into subsamples in the following way; Training sample: 50%, Validation sample: 30%, Test sample: 20%. The combination of the training and validation sample is called the development sample, which is used for training the model after the

hyperparameters have been determined.

## 5.2 Linear algorithms: Lasso & Ridge

We use the training sample estimate the model in 5.1 using the Lasso and Ridge methods with different $\lambda$ values ranging from 0,0001 to 10.000. For each $\lambda$ the model is evaluated on it's ability to predict the validation data by calculating the MSE. It is of great importance not to use the test data to determine the optimal hyperparameters, since this corresponds to using future information for predicting the future. For each of the models (Lasso and Ridge) the optimal $\lambda$ is determined, and used to train a model on the entire development data sample. With this new 'optimal' model we predict out-of-sample, that is, we predict the test data, and the MSE between the predictions and the actual data is calculated again.
As a robustness test we predict the test sample for 10 different models, where we change the degrees of polynomials allowed for the Lasso and the Ridge respectively. The results are presented in table 1.

Table 1: Results from out-of-sample predictions of different models estimated by the Lasso and Ridge methods and allowing for different degrees of polynomials.

| | Lasso | | Ridge | |
|---|---|---|---|---|
| Degrees | $MSE$ | $RMSE$ | $MSE$ | $RMSE$ |
| 1 | **0,027** | **0,164** | 0.028 | 0,167 |
| 2 | 0,029 | 0,169 | **0,027** | **0,164** |
| 3 | 0,029 | 0,170 | 0,028 | 0,167 |
| 4 | 0,029 | 0,169 | 0,028 | 0,167 |
| (...) | | | | |
| 10 | 0,029 | 0,172 | 0,030 | 0,174 |

Note: Degrees indicates the order of polynomials we allow, when the model fit is made.

We find, that our preferred models estimated using Lasso and Ridge both predict the out-of-sample data with a Root Mean Squared Error (RMSE) of 0,164. A point to be made from table 1 is how the RMSE increases in the degrees of polynomial features allowed. This can be due to

the curse of overfitting of the development sample, which weakens the predictive power, when the model is presented to unknown data. Thus, even though we use regularization methods to prevent overfitting, it seems that we are not able to avoid the problem completely. We will now continue to the more complex and non-linear algorithms with the purpose of constructing a model, that can perform even better predictions out-of-sample.

## 5.3 Non-linear algorithms: Decision Trees and Random Forests

The basic process of using a random forest algorithm is similar to other supervised machine learning algorithms. We set up our model 'RandomForestRegressor' and its hyperparameters. The model is then fitted to the training data, and the fitted model is then applied to the test data to obtain predictions. For performance evalution our predictions are compared to the actual test data. The process for Decision Tree Regressions is similar, although there is only one tree.

## 5.4 Optimizing hyperparameters of Decision Tree and Random Forests

We use the python module RandomizedSearchCV to optimize over all the hyperparameters listed. Each candidate in the randomized search takes a random integer betweem 3-12 for max depth, a random integer for max features between 1-5 and so on for the remaning parameters (min samples split: 2-11, min samples leaf: 1-6, bootstrap: True/ False and n estimators: 25,501).

The RandomizedSearchCV estimates the model on the training data using the Decision Tree and Random Forest method, respectively, for random combinations of hyperparameters. We iterate through 1.000 different combinations using the Decision Tree and 10.000 combinations using the Random Forest. The models are ranked by their validation score, provided by the RandomizedSearchCV module, and the three models with highest score, for both Decision Tree and Random Forest, are presented in table 2.

Table 2: Results from optimizing Decision Tree and Random Forest models with different hyperparameter sets

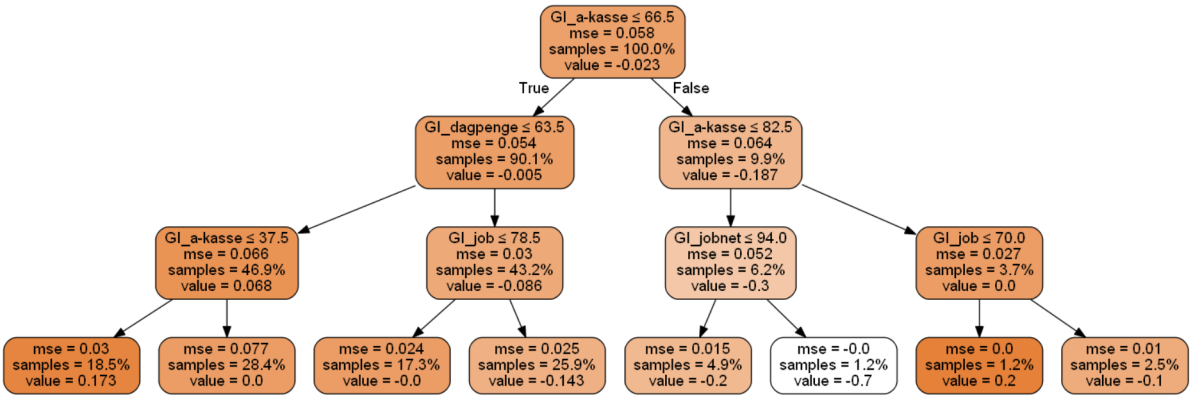
| | Decision Tree | | | Random Forest | | |
|---|---|---|---|---|---|---|
| Hyperparameters | I | II | III | **I** | II | III |
| max depth | **3** | 3 | 3 | **9** | 7 | 6 |
| max features | **1** | 4 | 4 | **2** | 2 | 2 |
| min samples leaf | **1** | 5 | 5 | **3** | 3 | 3 |
| min samples split | **2** | 5 | 8 | **8** | 2 | 3 |
| n estimators | **-** | - | - | **26** | 35 | 38 |
| bootstrap | **-** | - | - | **True** | True | True |
| Validation score | **0,074** | 0,057 | 0,057 | **0,172** | 0,170 | 0,161 |
| $MSE$ | **0,032** | 0,035 | 0,035 | **0,025** | 0,026 | 0,027 |
| $RMSE$ | **0,178** | 0,188 | 0,188 | **0,159** | 0,161 | 0,163 |

Note: The validation score is provided in the RandomizedSearchCV module.

Our preferred models estimated by Decision Tree and Random Forest yields a RMSE of 0,178 and 0,159 respectively. This imply, that our prefered non-linear models predict out-of-sample with an average error of 0,178 percentage points.

In Figure 4 below we have plottet the most optimal decision tree. It clearly shows the optimal maximum depth of 3, and that the first optimal split is the feature "A-kasse" being smaller or equal to 66.5. This split the data in two samples of 90.1 pct. and 9.9 pct., respectively. This way of splitting continues through the tree structure, and we end with a data split in eight samples, largest one containing 18.5 pct. of the data.

Figure 4: Decision Tree



# 6 Results

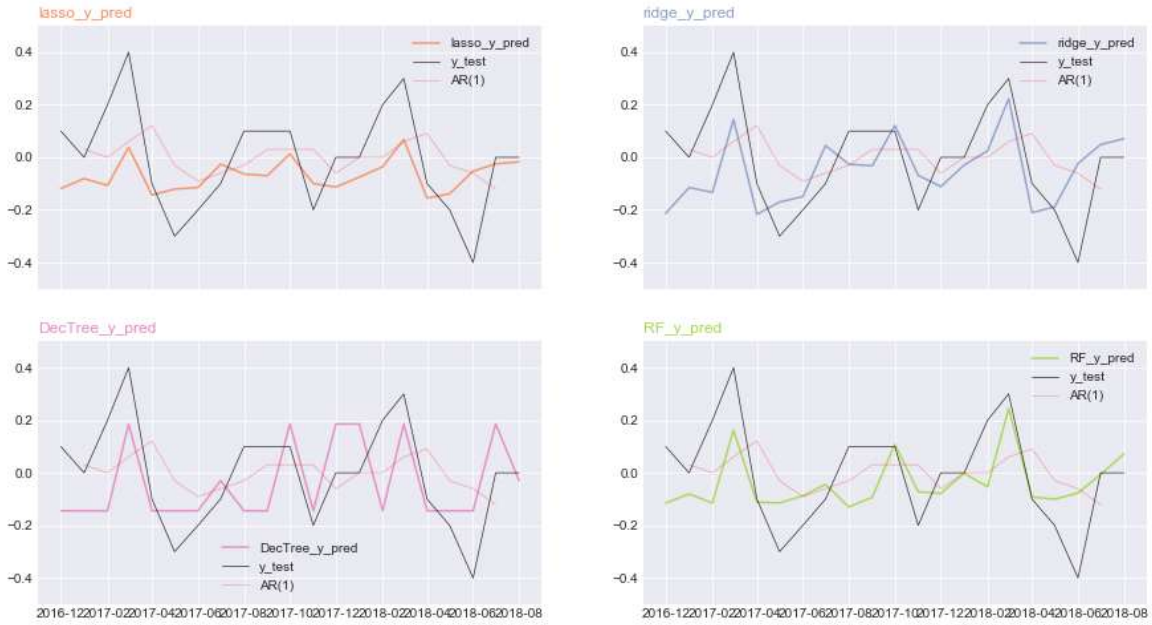
In Table 3 we present the result from our the best performing model variants of the Lasso, the Ridge, the Decision Tree and the Random Forest respectively. These models are marked in **bold** in Table 1 and Table 2. We also make a comparison to our benchmark, the AR(1) model estimated by OLS.

Table 3: Best models and deviation from AR(1) benchmark

| | **AR(1)** | **Lasso** | **Ridge** | **Decision Tree** | **Random Forest** |
|---|---|---|---|---|---|
| $MSE$ | 0,039 | 0,027 | 0,027 | 0,032 | 0,025 |
| $RMSE$ | 0,197 | 0,164 | 0,164 | 0,178 | 0,159 |
| **Deviation from AR(1)** | | | | | |
| $MSE$ | 1 | 0,692 | 0,692 | 0,821 | 0,641 |
| $RMSE$ | 1 | 0,832 | 0,832 | 0,904 | 0,807 |

We see that all of our preferred models estimated by Machine Learning algorithms performs better than the benchmark. The RMSEs indicate, that the Lasso and the Ridge both perform 16.8 pct. better as predictors of the test data. The Decision Tree is 9.6 pct. better, while the model estimated by Random Forest provides the best prediction with a RMSE 19.3 pct. lower than the benchmark. In Figure 5 below, we have visualized the four best performing models of how they fit the test data compared to the AR(1) model.

Figure 5: Best models visualized




# 7 Discussion

In this section we compare our results to the literature and discuss further possibilities with Machine Learning.

Our results contributes very well to the literature of correlation between Google search queries and unemployment and prediction of the last. Like Gonzalez-Fernandez & Gonzalez-Velasco (2018) we find that models including Google searches are better predictors for unemployment than just a simple AR(1) model. Further, our Random Forest algorithm is a stronger predictor than the econometric model in their analysis.

Same goes when comparing our results to the results from D'Amuri & Marcucci (2017). They found that when predicting one period ahead, including GI would give a 18 pct. better prediction. So again our Random Forrest algorithm seems to be a really good predictor for the Danish unemployment rate.

## 7.1 Cross-validation of Time Series

This is arguably the weakest link of this study and the first place we could reinforce our analysis if we were to extend this study. We perform a simple fixed-origin evaluation, that is, we split up our data in a way that keeps the inherent serial correlation in time series - but we only perform one split. Using only one split does not yield a high level of robustness. That could be improved with multiple splits i.e. a rolling-origin-update evaluation.

Both techniques still exhibit at least one flaw: First, a possible, but not necessary problem of premises from early periods becoming invalid in the future. A machine that continuously learns from the future might improve over time, but remains unable to unlearn its old ways. A walk forward analysis can resolve this problem, but was outside the scope of this project.

## 7.2 Performance measures

We have only used one performance measure in this paper, i.e. the MSE and its sibling RMSE. There are other possible measures we coiuld have used namely the Mean Absolute Error (MAE), which can be defined as the average magnitude of the errors in a set of forecasts. It measures accuracy for continuous variables."[7] There is a wide range of papers arguing which measure is better to use and in which domain.

The main difference between MSE and MAE is that MAE treats all the errors equally whereas MSE squares each error before taking the mean. This way, MSE gives higher importance to large errors. In addition, MAE takes the absolute value of the predicted and observed values, neglecting the direction of the errors. This is different from MSE that does not take the absolute value of errors. Considering the above mentioned arguments, we decided that MSE is a better adapted measure for our analysis. The MSE however cannot be interpreted directly in relation to the data, and therefore we take use the RMSE to be more able to interpret.

## 7.3 Limitations

### Number of lags

We have only considered one lag, but a natural extension would be to consider more lags. If we look at the dynamics of the Danish economy, the unemployment rate usually maintains its

[7] http://www.eumetrain.org/data/4/451/english/msg/ver$_c ont_v ar/uos3/uos3_k o1.htm$

direction for a number of periods before shifting. Therefore, we do not before deem it impossible that including more lags would increase prediction performance.

**Feature importance**

Taking a more thorough look at the set of keywords might also yield increased performance. The lowest hanging fruit would be to exclude the word *job* as it does not seem to correlate with unemployment at all. It could also be interesting to analyse increase the number of lags on the search terms. In a period of recession or foreboding lay-offs, workers might start to look for jobs multiple months in advance.

**Hyperparameterisation**

Due to computational limitations we chose to run Randomized Searches in order to find optimal hyperparameter values for our Decision Tree and Random Forest. An alternative, a Grid Search, is more thorough, but also vastly more demanding in terms of computational power. Using Randomized Search we ran 10.000 iterations on two computers, and in spite of the high number of iterations, our two results were still different.

We also chose to use a seed to generate random numbers, in order to preserve reproducibility, but as we played around with different seeds we realized our results were not completely robust. A Grid Search should mitigate this problem.

# 8 Concluding remarks

The aim of this paper was to analyze if job-related Google searches might help forecast the change in the unemployment rate in Denmark one month ahead.

We provided this analysis in several steps. First, we selected the relevant keywords and scraped the data from Google Index and Statistics Denmark. Next, we provided an empirical analyses using the AR(1), Lasso and Ridge methods and we compared the outcomes for choosing the best model fit. At the end, we applied two non-linear algorithms, Decision tree and Random forest and we discussed the results. It can be noticed that GI seems to have significant relevance in predicting the change in the unemployment rate. We believe that these results are convincing and internet searches should be used for future unemployment forecasting.

# References

Bergmeir, C., & Benitez, J. M. 2012. "On the use of cross-validation for time series predictor evaluation". *Information Sciences*, 192–213.

Bergmeir, C., Hyndman, R. J., & Koo, B. 2018. "A note on the validity of cross-validation for evaluating autoregressive time series prediction". *Statistics  Data Analysis*, 70–83.

D'Amuri, Francesco, & Marcucci, Juri. 2017. "The predictive power of Google searches in forecasting US unemployment". *International Journal of Forecasting*, 801–816.

Gonzalez-Fernandez, Marcos, & Gonzalez-Velasco, Carmen. 2018. "Can Google econometrics predict unemployment? Evidence from Spain". *Economic Letters*, 42–45.

Raschka, Sebastian, & Mirjalili, Vahid. 2017. *Python Machine Learning*. Second edn. Packt Publishing.

Tashman, L.J. 2000. "Out-of-sample tests of forecasting accuracy: an analysis and review". *International Journal of Forecasting*, 437–450.