Predict Future Sales*

Fazel Arasteh[1] and Ali Nematichari[2]

***Abstract*— Sales prediction is concerned with estimating future sales of companies in the industry. Accurate short-term sales prediction allows companies to minimize stocked and expired products inside stores and at the same time avoid missing sales. This paper compares and reviews several existing methods for sales prediction. In this paper, we apply some data preprocessing and feature engineering for the time series prediction. We discuss important design decisions of a data analyst such as the input variables to use for predicting sales. Finally we compare the results of existing methods on a Kaggle competition task named "predict future sales". Note that this competition is currently open.**

I. INTRODUCTION

In todays highly competitive and constantly changing business environment, the accurate and timely estimation of future sales, also known as sales prediction or sales forecasting, can offer critical knowledge to companies involved in the manufacturing, wholesale or retail of products. Short-term predictions mainly help in production planning and stock management, while long-term predictions can help in business development decision making.[2] Sales prediction is important in every industry. For instance in the food industry, due to the short shelf-life of many of the products, which leads to loss of income in both shortage and surplus situations, the sales prediction is critical. Ordering too many leads to waste of products, while ordering too few leads to customer loss. Moreover, food consumer demand is constantly fluctuating due to factors such as price, promotions, changing consumer preferences or weather changes. [9] Sales prediction is typically done arbitrarily by managers. However, skilled managers are hard to find and they are not always available (e.g. they may get sick or take a leave). Therefore, sales prediction should be supported by computer systems that can play the role of a skilled manager when she is not there and/or help her take the right decision by providing estimates of future sales. One way to build such a system would be to try and model the expert knowledge of skilled managers within a computer system. Alternatively, one can exploit the wealth of sales data and related information to automatically construct accurate sales prediction models via machine learning techniques. The latter is a much simpler process, it is not biased from the particularities of a specific sales manager and it is dynamic, meaning it can adapt to changes in the data. Furthermore, it has the potential to outweigh the prediction accuracy of a human expert, who typically is imperfect. [8] Sales prediction is a time series forecasting task. Classical statistical techniques, such as autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) can be used to tackle this task. Also, taking a machine learning approach to deal with a time series forecasting can be more powerful and flexible. Powerful, because it allows the use of modern state-of-the-art supervised learning algorithms for regression [6]. Flexible, because it allows the inclusion of additional useful input variables, external to the given time-series. This paper reviews review classical and machine learning techniques for sales prediction, an important task for a range of companies, such as supermarkets, restaurants, bakeries and patisseries. To this end, we aim to use a Kaggle competition data-set-Predict Future Sales. In section II we overview the data-set and the data preprocessings that we took. In section III we overview the seasonality analysis which is an important analysis in the prediction of time series related to sales. In section IV we go through the model selection techniques for time series data. Section V covers an overview of the models that we compare. In section VI we present the results.

II. DATA-SET DESCRIPTION AND DATA PREPROCESSING

The data-set is borrowed from the "future sales competition" on Kaggle. The data-set contains historical data on the number of items sold from each item in each store from an 2013 to Oct 2015. Each entry in the data-set contains a date column, shop-id, item-id, item-price,and number of the items sold. Each item also belongs to an item category. The task is to predict the total count of sales in the next month (Oct 2015) for each item-id and shop-id pair. There are about 22K items, 60 shops, and 84 item-categories in the data-set. For every shop-id and item-id pair there exist a time-series. Hence, there are 720K time series for all the shop-id and item-id pairs. As a result, it is not feasible to generate all these time series before head. Instead, we build the time series on demand. For instance, if we want to know the sales for item-id=X and shop-id=Y, then we search the data-set once and create the time-series for that pair. Based on the generated time series and the selected prediction model we predict the future sales.

A. Data Preprocessing

1) Daily to monthly: The original data is stored by date. However, we are interested in monthly number of sales. Hence, we have to extract the monthly data from the data-set by aggregating the daily sales data.

* Data Mining Course,Fall 2019, Prof. Aijun Ann

[1] EECS, Lassonde School of Engineering, York University, Toronto, Ontario, Canada fazelara at eecs.yorku.ca

[2] EECS, Lassonde School of Engineering, York University, Toronto, Ontario, Canada alinemat at eecs.yorku.ca

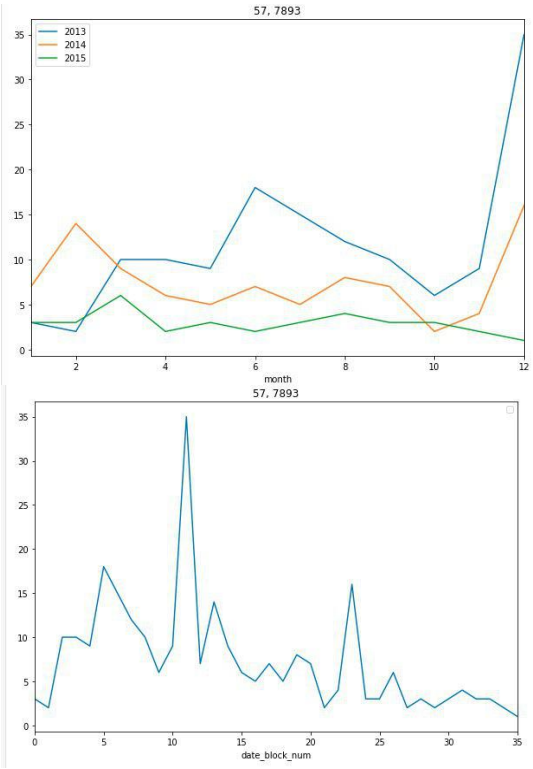

Fig. 1. Sales of the item 7893 in shop 57

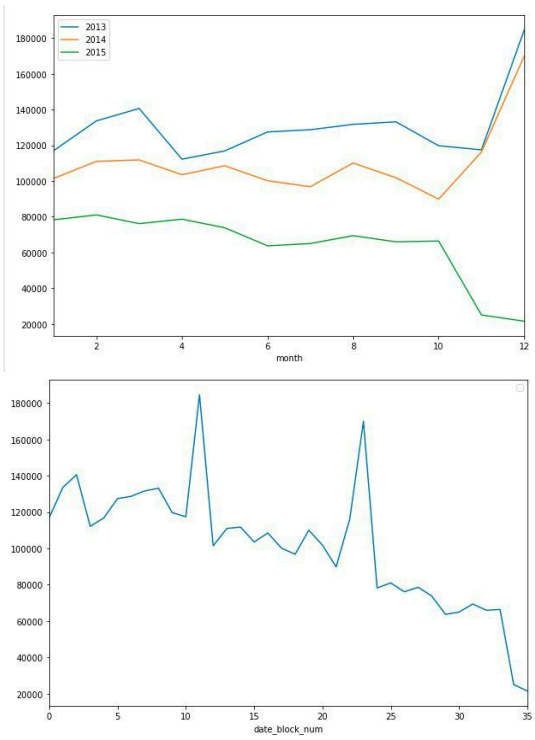

Fig. 2. Total Sales

2) Wrong Data: In the original data-set there is a column named "date-block-number". This column is supposed to show a consecutive month number, used for convenience. For instance, January 2013 is 0, February 2013 is 1,..., October 2015 is 33. One can use this column to extract monthly sum of sales from the daily sales. However, surprisingly (or intentionally!) this column doesn't match with the date column. We used the date column to reassign the values in the date-block-number column. Moreover, according to the website of the competition, the data is from Jan 2013 to Oct 2015 (less than 3 years). However, we found some data that dated to Nov and Dec 2015. We treat the data of the Nov and Dec 2015 as invalid data and removed them from the data-set.

3) Add Zeros: When we extract the time series for an item-id and shop-id pair, some items do not have any sales in some months. We need to add the value of zero sales for such months in the time series.

4) Feature generation: We can treat the problem of time series predicting as a machine learning regression task. However, to this end, we need to generate promising features out of the time series. We discuss these features in the regression method section (section V subsection B).

B. Data Visualisation

In this section, we illustrate the figure of time series for some sample items. Fig.1 illustrates the time series for the sales of shop-id= 57 and item-id=7893. Fig.2 illustrates the time series for the sales of all items in the data set during the time.

III. Seasonality Analysis

Seasonality is a cycle that repeats over time, such as monthly or yearly. This repeating cycle may obscure the signal that we wish to model when forecasting, and in turn may provide a strong signal to our predictive models. A repeating pattern within each year is known as seasonal variation, although the term is applied more generally to repeating patterns within any fixed period. A cycle structure in a time series may or may not be seasonal. If it consistently repeats at the same frequency, it is seasonal, otherwise it is not seasonal and is called a cycle. Identifying and removing the seasonal component from the time series can result in a clearer relationship between input and output variables.[1] The top figure in Fig.1 and in Fig.2 illustrate the seasonality effect. Note that according to the Kaggle competition website, the data for the last two months are not valid, and that is probably the reason that we see that the last two months of 2015 are not following the seasonality. Fig.3 illustrates the polar figure for the sales of the item7893 in shop57. Based on these figures we realize that we have seasonality in the data. In the next section we discuss the seasonal decomposition.

A. Seasonal decomposition

The model of seasonality can be removed from the time series. This process is called Seasonal Adjustment, or Deseasonalizing. Often this is done to help improve understanding of the time series, but it can also be used to improve forecast accuracy. A time series where the seasonal component has been removed is called seasonal stationary.

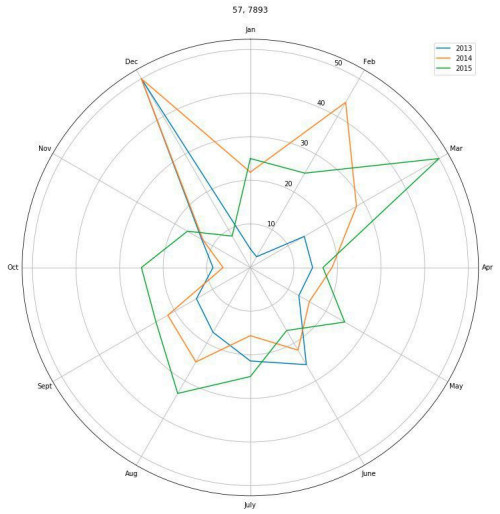

Fig. 3. Total Sales of item7893 in shop 57

A time series with a clear seasonal component is referred to as non-stationary [1]. The classical decomposition method originated in the 1920s. It is a relatively simple procedure, and forms the starting point for most other methods of time series decomposition. In classical decomposition, we assume that the seasonal component is constant from year to year. There are two forms of classical decomposition: an additive decomposition and a multiplicative decomposition. If we assume an additive decomposition, then we can write:

$$y_t = S_t + T_t + R_t$$

where y_t is the data, S_t is the seasonal component, T_t is the trend-cycle component, and R_t is the remainder component, all at period t. Alternatively, a multiplicative decomposition would be written as

$$y_t = S_t * T_t * R_t$$

The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate. Multiplicative decompositions are common with economic time series. An alternative to using a multiplicative decomposition is to first transform the data until the variation in the series appears to be stable over time, then use an additive decomposition. When a log transformation has been used, this is equivalent to using a multiplicative decomposition because $y_t = S_t * T_t * R_t$ is equivalent to $logy_t = logS_t * logT_t * logR_t$ [5]. If the seasonal component is removed from the original data, the resulting values are the seasonally adjusted data. For an additive decomposition, the seasonally adjusted data are given by $y_t - S_t$, and for multiplicative data, the seasonally adjusted values are obtained using y_t/S_t.

Moving average smoothing or Rolling Mean: A moving average of order m can be written as:

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}$$

where m=2k+1. That is, the estimate of the trend-cycle at time t is obtained by averaging values of the time series within k periods of t. Observations that are nearby in time are also likely to be close in value. Therefore, the average eliminates some of the randomness in the data, leaving a smooth trend-cycle component. We call this an m-MA, meaning a moving average of order m. Additive Decomposition:

- Estimating trend T(t) through a rolling mean
- Calculate the detrended series: $y_t - \hat{T}_t$
- To estimate the seasonal component for each season, simply average the detrended values for that season. For example, with monthly data, the seasonal component for March is the average of all the detrended March values in the data. These seasonal component values are then adjusted to ensure that they add to zero. The seasonal component is obtained by stringing together these monthly values, and then replicating the sequence for each year of data. This gives $\hat{S}_t$
- The remainder component is calculated by subtracting the estimated seasonal and trend-cycle components: $\hat{R}_t = \hat{y}_t - \hat{T}_t - \hat{S}_t$

Multiplicative decomposition: A classical multiplicative decomposition is similar, except that the subtractions are replaced by divisions. Fig.4 Illustrates the additive seasonal decomposition for time series of item 22087 in shop 59.

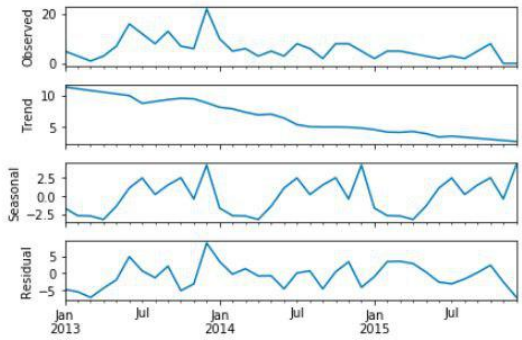

Fig. 4. Additive Seasonal Decomposition

One way to use the decomposition for forecasting purposes is the following:

- Decompose the training time series with some decomposition algorithm.

$$Y(t) = S(t) + T(t) + R(t)$$

- Compute the seasonally adjusted time series Y(t)-S(t). Use any model you like to forecast the evolution of the seasonally adjusted time series.
- Add to the forecasts the seasonality of the last time period in the time series (in our case, the fitted S(t) for last year).

IV. Time series model selection

Cross-validation (CV) is a popular technique for tuning hyperparameters and producing robust measurements of model performance. Two of the most common types of cross-validation are k-fold cross-validation and hold-out cross-validation. First, we split the dataset into a subset called the training set, and another subset called the test set. If any parameters need to be tuned, we split the training set into a training subset and a validation set. The model is trained on the training subset and the parameters that minimize error on the validation set are chosen. Finally, the model is trained on the full training set using the chosen parameters, and the error on the test set is recorded. With time series data, particular care must be taken in splitting the data in order to prevent data leakage. In order to accurately simulate the real world forecasting environment, in which we stand in the present and forecast the future, the forecaster must withhold all data about events that occur chronologically after the events used for fitting the model [7]. So, rather than using k-fold cross-

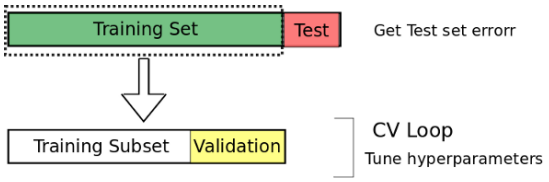

Fig. 5. Cross Validation

validation, for time series data we utilize hold-out cross-validation where a subset of the data (split temporally) is reserved for validating the model performance. For example, see Fig.5 where the test set data comes chronologically after the training set. Similarly, the validation set comes chronologically after the training subset. However, this requires that you have enough data to set aside a test set and still have data to build a model with. But time series data is often very small compared to the data sets used in image processing or NLP. Two years of weekly sales data for a product at a given location is only 104 data points (barely enough to capture any seasonality). A Quarterly economic indicator data, taken over 10 years is only 40 data points. Hence, we resort to using information criteria such as the AIC, the AICc or the BIC. These are model selection metrics, which essentially try to approximate the test step analytically. The idea is that we dont have an empirical way of determining the generalization error of a model, but we can estimate this error by using information theoretical considerations [4]. In practice, we train a predetermined set of models and then select the one that has the lowest AIC or BIC. Besides allowing for training models with limited data, using such criteria for model selection is very convenient when we want to automate forecast generation. For example in retail it is not unusual that we have to generate forecasts for millions of individual time series a large retailer will carry 20K to 30K products in several hundred locations, resulting in millions individual times series (one for each product/location combination). The Akaike Information Criterion (AIC) lets you test how well your model fits the data set without over-fitting it. The model with the lower AIC score is expected to strike a superior balance between its ability to fit the data set and its ability to avoid over-fitting the data set. The formula for the AIC score is as follows:

$$AIC = 2k - 2\ln(\mathcal{L})$$

where k is the number of the model parameters and $\mathcal{L}$ is the maximum value of the likelihood function of the model [4].

V. Models Description

In this section we overview some promising models for time series predictino. In general these methods are divided into two categories. First, the theoretical methods that have a basis in signal processing and do not require a feature vector. Second, are the machine learning methods. In order to transform the problem into a machine learning regression task, we have to generate a training-set out of the time series.

A. Theoritical Methods

1) Naive, SNaive: In the Naive model, the forecasts for every horizon correspond to the last observed value.

$$\hat{Y}(t+h|t) = Y(t)$$

An extension of the Naive model is given by the SNaive (Seasonal Naive) model. Assuming that the time series has a seasonal component and that the period of the seasonality is T, the forecasts given by the SNaive model are given by:

$$\hat{Y}(t+h|t) = Y(t+h-T)$$

Therefore the forecasts for the following T time steps are equal to the previous T time steps. In our application, the SNaive forecast for the next year is equal to the last years observations.These models are often used as benchmark models.

2) Exponential smoothing: Exponential smoothing is one of the most successful classical forecasting methods. In its basic form it is called simple exponential smoothing and its forecasts are given by:

$$\hat{Y}(t+h|t) = \alpha y(t) + \alpha(1-\alpha)y(t-1) + \alpha(1-\alpha)^2 y(t-2) + ...$$

with $0 < \alpha < 1$ We can see that forecasts are equal to a weighted average of past observations and the corresponding weights decrease exponentially as we go back in time. [5]

3) ARIMA, SARIMA: ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting, and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

A stationary time series is one whose properties do not depend on the time at which the series is observed.14 Thus, time series with trends, or with seasonality, are not stationary the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise

series is stationary it does not matter when you observe it, it should look much the same at any point in time. In general, a stationary time series will have no predictable patterns in the long-term. One way to make a non-stationary time series stationary is to compute the differences between consecutive observations. This is known as differencing. A seasonal difference is the difference between an observation and the previous observation from the same season. So

$$y_t^{'} = y_t - y_{t-m}$$

where m is the number of seasons. These are also called lag-m differences", as we subtract the observation after a lag of m periods. If seasonally differenced data appear to be white noise, then an appropriate model for the original data is

$$y_t = y_{t-m} + \epsilon_t$$

Forecasts from this model are equal to the last observation from the relevant season. That is, this model gives seasonal naive forecasts.

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order p can be writen as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t$$

where ϵ_t is white noise. This is like a multiple regression but with lagged values of y_t as predictors. We refer to this as an AR(p) model, an auto-regressive model of order p. Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q}$$

Where ϵ_t is white noise. We refer to this as an MA(q) model, a moving average model of order q. Of course, we do not observe the values of ϵ_t , so it is not really a regression in the usual sense. Notice that each value of y_t can be thought of as a weighted moving average of the past few forecast errors. However, moving average models should not be confused with the moving average smoothing. A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values. If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average (in this context, integration is the reverse of differencing). The full model can be written as

$$y_t^{'} = c + \phi_1 y_{t-1}^{'} + ... + \phi_p y_{t-p}^{'} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$

where $y_t^{'}$ is the differenced series (it may have been differenced more than once). The predictors on the right hand side include both lagged values of y_t and lagged errors. We call this an ARIMA(p,d,q) model, where

- p=order of the autoregressive part.
- d=degree of first differencing involved.
- q=order of the moving average part.

The SARIMA model (Seasonal ARIMA) extends the ARIMA by adding a linear combination of seasonal past values and/or forecast errors. For a complete introduction to ARIMA and SARIMA models, we cite chapter 8 in [5]

B. Machine Learning Methods

The basic concept is that we forecast the time series of interest y assuming that it has a linear relationship with other time series x. For example, we might wish to forecast monthly sales y using total advertising spend x as a predictor. Or we might forecast daily demand y using price x_1 and the day of week x_2 as predictors. The forecast variable y is sometimes also called the regressand, dependent or explained variable. The predictor variables x are sometimes also called the regressors, independent or explanatory variables. [5] Perhaps the most important factor affecting the success or failure of a machine learning project is the features used [3]. A study of the features that have been used in past sales prediction projects is therefore of primary importance. Predictive variables can be categorized into sales related features that can be obtained from the companys data warehouse (internal) and features obtained from external sources.[10] The most common type of internal features are lagged variables, i.e. product sales figures for past time units (days, weeks, etc). Lagged variables are the main mechanism by which the relationship between past and current values of a series can be captured by propositional learning algorithms. They create a window or snapshot over a time period. Essentially, the number of lagged variables created determines the size of the window. An appropriate size for the window can be determined through experimentation. Reasonable default values can be determined based on the time granularity of the given problem. For example, if you had monthly sales data then including lags up to 12 time steps into the past would make sense; for hourly data, you might want lags up to 24 time steps or perhaps 12. The window is not necessary to include consecutive time units, i.e. for weekly data it could make sense to include the weeks of the previous 3 months, but also, exactly the same week 1 year ago. In addition, one could average consecutive lagged variables into a single field, in order to reduce the number of input variables, as large numbers of input variables can have an adverse effect on some learning algorithms. [8] An important class of internal features are product-related features. These typically include the products brand, its packaging information, whether it is on promotion, its price elasticity, whether it has a short expiration date and whether it is a holiday product. External features used for sales prediction can be weather related data, financial indicators and date-related features such as holidays or events drawing mass consumption.[8] The following internal features were considered in [2]: The sales of the previous 6 days, the sales

of the corresponding day of the last year (same day of the week, approximately same date), the sales of the previous 6 days of that day and the percentile change in sales between the current year and the previous year. The 5 features that were found to be more useful empirically were the sales of this year with lag 1 (previous day) and 6 (the same day of the previous week, as outlets were open 6 days per week) and the corresponding sales of the previous year with lag 3, 5 and 6. Accordingly, we designed our feature vector is as follows:

- We use a window of 3 previous months of sales.
- To account for the seasonality effect, we bring the last year value of sales in the same month as a feature. However, for the first year data, we placed the last month value as last year value. Hence the last month value is duplicated in the feature vector for the data of the first year.
- The price of the item in the previous month can also be very important. For instance if the company decides to put the item on sale, we expect more sales for it on the next month.
- The amount of sales of the category that the item belongs to can also help in prediction. We consider the last three months and last year value of the sales for the item category as features as well.

1) Polynomial Regression: One of the problems with time series prediction is that the amount of the training set is very limited. Hence, any model that needs many training examples to converge is doomed to fail. Polynomial Regression models, however, can have a moderate complexity and hence, are promising for this purpose.

2) Multi Layer Percepteron: Neural Networks due to their abilities to fit to complicated datas are always a candidate for testing. However, in the case of time series, the amount of training data is not abundant. As a result, the NN may not be able to converge. Here we experiment on MLP for the sake of the comparison.

3) LSTM: LSTM models can be used to forecast time series (as well as other Recurrent Neural Networks). LSTM is an acronym that stands for Long-Short Term Memories. This technique allows to keep tracks of dependencies of new observations with past ones. LSTMs may benefit from transfer learning techniques even when applied to standard time series. However, they are mostly used with unstructured data (e.g. audio, text, video). We use the same feature vector as the previous section.

VI. Results

To compare the models we use RMSE criteria. Due to the limitations in Kaggle's policy of submission which only allows 5 submissions per day for a team, we can not use the original test set of the competition for our comparison. Consequently, we created a separate test-set. To this end, we took the data of 33th month as the test data. Also the original test set is too big (220K entry). We moved to using a much smaller test set of 100 time series randomly selected from the original test-set. As a result, the value of the obtained RMSE is not comparable with the Kaggle leader board. Table. I compares the RMSE for different algorithms.

Algorithm	RMSE
Naive	204
SNaive	222
Exponential Smoothing	204
Arima	204
SArima	213
Polynomial Regression (degree=1, linear)	154
MLP	136
LSTM	NOT AVAIL

TABLE I
RMSE FOR 33TH MONTH PREDICTIONS

Although we reduced the test set greatly, it is still too big for LSTM because LSTM is comparatively much more time and memory consuming. (We ran out of memory training 100 LSTM models). Instead we reduced the test set size to 10 randomly selected time series to compare LSTM with other models. Table. II illustrate the results. We can see that LSTM is performing better than all models but it is not feasible for training thousands of time series.

Algorithm	RMSE
MLP	18
LSTM	7

TABLE II
RMSE FOR 33TH MONTH PREDICTIONS

VII. Conclusions

In this paper we tackled one of the most challenging problems in data science, prediction of the time series. We used a data-set from a Kaggle competition about sales prediction. We first described and visualized the data. Then we discussed different methods of dealing with time series prediction. We experimented the different methods on the problem and compared the result. According to our results, for this especial task the LSTM model performs best. However LSTM is too computationally heavy and not feasible for thousonds of time series.

References

[1] Paul Cowpertwait and Andrew Metcalfe. "Introductory Time Series With R". In: Jan. 2009. ISBN: 0387886974, 9780387886978. DOI: 10.1007/978-0-387-88698-5.

[2] Philip Doganis et al. "Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing". In: *Journal of Food Engineering* 75.2 (2006), pp. 196–204.

[3] Pedro M Domingos. "A few useful things to know about machine learning." In: *Commun. acm* 55.10 (2012), pp. 78–87.

[4] Trevor Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Jan. 2009, pp. 220–230. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.

[5] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[6] Niels Landwehr, Mark Hall, and Eibe Frank. "Logistic model trees". In: *Machine learning* 59.1-2 (2005), pp. 161–205.

[7] Leonard J Tashman. "Out-of-sample tests of forecasting accuracy: an analysis and review". In: *International journal of forecasting* 16.4 (2000), pp. 437–450.

[8] Grigorios Tsoumakas. "A survey of machine learning techniques for food sales prediction". In: *Artificial Intelligence Review* 52.1 (2019), pp. 441–447.

[9] Jack GAJ van der Vorst et al. "Supply chain management in food chains: Improving performance by reducing uncertainty". In: *International Transactions in Operational Research* 5.6 (1998), pp. 487–499.

[10] Indre Žliobaite, Jorn Bakker, and Mykola Pechenizkiy. "Towards context aware food sales prediction". In: *2009 IEEE International Conference on Data Mining Workshops*. IEEE. 2009, pp. 94–99.