CIS 422/522

SOFTWARE METHODOLOGIES I

PROJECT 1

TIME SERIES ANALYSIS SUPPORT FOR DATA SCIENTISTS

Juan Flores Jan. 5, 2021

1. OVERVIEW

Artificial Intelligence (AI) originated in 1956, at a conference at Dartmouth College, in Hanover, New Hampshire. John McCarthy coined the term during that meeting. Colloquially, AI refers to machines that mimic cognitive functions, such as learning and problem-solving.

Machine learning (ML) is the study of computer algorithms that improve automatically through experience (i.e., they learn.); it is considered a subset of AI. ML algorithms build a model based on sample data training data). ML models make predictions or decisions without being explicitly programmed to do so. One subarea of ML is the study of Artificial Neural Networks.

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from many structural and unstructured data. Data science deals with the application of AI, ML, and Data Mining to different fields of Science and human activities.

Time series forecasting is an important task that derives forecasting models using tools from Data Science and ML. Fig 1. shows the relationship between those and other related fields.

Time series forecasting is an important task in the fields of Data Science and Machine Learning. Time series originate from almost any area of human activity, and forecasting these time series provides a valuable insight into things that may happen in the future. Predictions of future outcomes and the underlying process to derive predictions can inform and influence the decision-making process. For instance, when generating renewable energy with solar panels, the two most important variables that impact the amount of energy produced by a solar cell are solar irradiance and ambient temperature. Each of these variables can be measured over time to create a time series. Solar power plant managers and engineers are interested in estimating how much energy the plant will produce in the near future. Fig.2. shows a solar plant and a time series of the total irradiance over a day, as measured by a pyranometer.

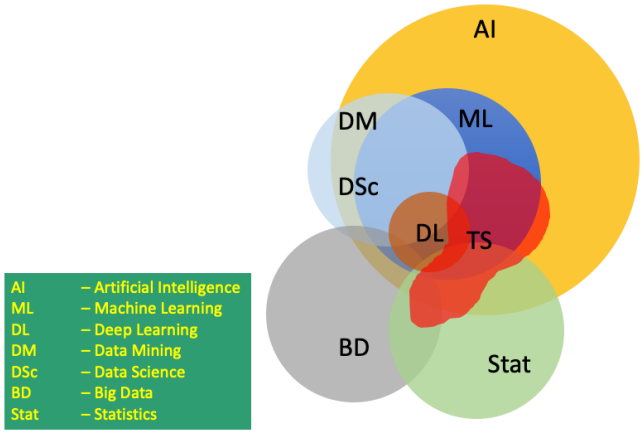

Fig 1. AI, ML, Time Series Forecasting, and Related Fields

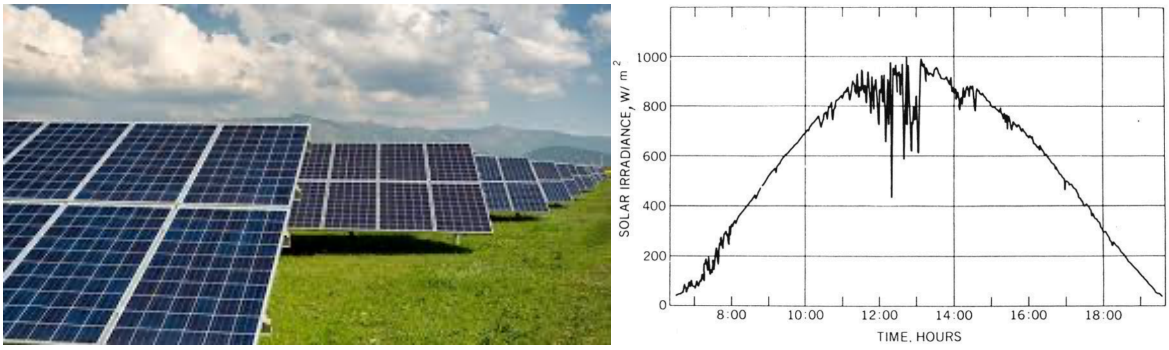

Fig. 2. Solar Energy Production and Solar Irradiance Time Series

A time series is the recording of the magnitude of a variable along with the time domain[1]. This recording happens at equally spaced instants of time, and no reading is missing at any time instant. Say that the variable of interest is x, and that recording starts at an initial time t_0, every Δ seconds, for N times. The time series X is a sequence of N elements.

$$X = \{x_0, x_1, \dots, x_k, \dots, x_N\} \quad (1)$$

where k is the time index and x_k is the magnitude of x at the instant $t_0 + (k-1)\Delta$.

The goal of forecasting is to determine an estimate for the value of $x_{N+\Delta}$. A forecasting model, F, is the implementation of a function that maps from the history of the time series to the forecasted value. See Montgomery et al. [1] for more information.

[1] There is the possibility of recording several related variables, which yields vector time series. We will only be concerned with scalar time series (i.e., those composed of a single variable).

$$F: X \rightarrow x_{N+\Delta} \quad (2)$$

Fig. 3 shows the results of a search for the term "Time Series Forecasting" in Google Trends, indicating how active the term is in web searches. The recording of this activity is a time series itself!

Fig. 3. Activity of the term "Time Series Forecasting," produced by Google Trends

The popularity of this area has grown following the success of Deep Learning methods. The amount of available data has grown with the popularity of IoT devices, installed everywhere.

2. TIME SERIES ANALYSIS

Statistical methods such as ARIMA, Exponential Smoothing, etc., have traditionally solved the forecasting problem [1, 2], Recently the forecasting problem has attracted the attention of the Machine Learning Community [3], and there is a variety of methods to produce time series forecasting models [4]. The development of better algorithms and more powerful hardware has enabled the development of Deep Learning techniques [5], which have impacted mainly the models produced by Artificial Neural Networks.

A Data Scientist (DS) receives a time series of the form given by (1) and tries to produce a model of the form in (2). This process is known as Time Series Analysis and Modeling. To that end, a DS has to use several preprocessing, data transformations, model hypothesis, training, validation, and visualization steps, among others.

The model formulation process is a non-linear search for the right sequence of steps to learn a data model. The Data Science community has developed many libraries to implement many of those steps. One of the most popular libraries in the Machine Learning field is Scikit-Learn [6]. Pandas is a widely used library for data processing, containing a module for Time Series Manipulation [7, 8]. The final result of this non-linear search is a pipeline (a composition) of operations on different kinds of data. Unfortunately, there does not exist a mathematical model or a straightforward procedure that takes a DS from (1) to (2).

This project aims to support a DS in this non-linear search of a sequence of transformations that produce a reasonable performance model. We cannot say the optimal model since the search space is too large to ensure optimality in the forecasting model space.

3. TIME SERIES TRANSFORMATIONS

You are to design the different objects/methods to represent time series, models, statistics, etc. Alternatively, you can use the objects already developed by Pandas, Scikit-Learn, or any other library of your choice. You will also need to search for the implementation of the different preprocessing and modeling steps in the available libraries.

The different time series transformations can be categorized into the following groups:

3.1. INPUT/OUTPUT

The file format to store a time series can be csv, pickle, parquet, or any other. The contents of such a file are couples of the form (time, magnitude). In pickle files, we directly store an object on a pickle file and read it from there.

- `read_from_file(input_file_name)`

- `write_to_file(output_file_name)`

3.2. PREPROCESSING

- `denoise(ts)` – Removes noise from a time series. Produces a time series with less noise than the original one. This function can be implemented using moving (or rolling) media or median (included in the Pandas library.)

- `impute_missing_data(ts)` – Missing data are often encoded as blanks, NaNs, or other placeholders. At this point, let us assume that a single point is missing, and it can be computed from its adjacent points in time. There are other ideas, for instance, those proposed by Kantz et al. [9].

- `impute_outliers(ts)` – Outliers are disparate data that we can treat as missing data. Use the same procedure as for missing data (sklearn implements outlier detection.) This function is better applied using the higher dimensional data produced by TS2DB (see below.)

- `longest_continuous_run(ts)` – Isolates the most extended portion of the time series without missing data. It returns a time series.

- `clip(ts, starting_date, final_date)` – clips the time series to the specified period's data.

- `assign_time(ts, start, increment)` – In many cases, we do not have the times associated with a sequence of readings. Start and increment represent t_0 Δ, respectively.

- `difference(ts)` – Produces a time series whose magnitudes are the differences between consecutive elements in the original time series.

- `scaling(ts)` – Produces a time series whose magnitudes are scaled so that the resulting magnitudes range in the interval [0,1].

- `standardize(ts)` – Produces a time series whose mean is 0 and variance is 1.

- `logarithm(ts)` – Produces a time series whose elements are the logarithm of the original elements.

- `cubic_root(ts)` – Produces a time series whose elements are the original elements' cubic root.

- `split_data(ts, perc_training, perc_valid, perc_test)` – Splits a time series into training, validation, and testing according to the given percentages.

- design_matrix(ts, input_index, output_index)

- design_matrix(ts, m_i, τ_i, m_o, τ_o)

The input index defines what part of the time series' history is designated to be the forecasting model's input. The forecasting task determines the output index, which indicates how many predictions are required and how distanced they are from each other (not necessarily a constant distance.)

- `ts2db(input_filename, perc_training, perc_valid, perc_test, input_index, output_index, output_file_name)` – this function combines reading a file, splitting the data, converting to database, and producing the training databases.

3.3. MODELING AND FORECASTING

The Scikit-Learn library includes most of the functions defined in this subsection. You may decide whether to implement wrappers, if necessary or to use them as they are. You do not need to be concerned, at this point, about the quality of the produced models.

- `mlp_model(input_dimension, output_dimension [, layers])` – defines an ANN Multi-Layer Perceptron model. You may use the same defaults as proposed by sklearn.

- mlp.fit(x_train, y_train) – trains the mlp model.

- `mlp.forecast(x)` – produces a forecast for the time series's current state, x.

You may include other regression models, e.g., Random Forest. In that case, you may want to include a set of functions similar to the ones described above.

- `rf_model()` – defines a Random Forest model. You may use the same defaults as proposed by sklearn.

- rf.fit(x_train, y_train) – trains the rf model.

- `rf.forecast(x)` – produces a forecast for the time series' current state, x.

3.4. STATISTICS AND VISUALIZATION

Feel free to use any existing plotting library. Matplotlib, Seaborn, and Bokeh are examples of many libraries you may want to use.

- `plot(ts | ts_list)` – Plots one or more time series. Adjust the time axis to display data according to their time indices.

- `histogram(ts)` – Compute and draw the histogram of the given time series. Plot the histogram vertically and side to side with a plot of the time series.

- `box_plot(ts)` – Produces a Box and Whiskers plot of the time series. This function also prints the 5-number summary of the data.

- `normality_test(ts)` – Performs a hypothesis test about normality on the time series data distribution. Besides the result of the statistical test, you may want to include a quantile plot of the data. Scipy contains the Shapiro-Wilkinson and other normality tests; matplotlib implements a qqplot function.

- `mse(y_test, y_forecast)` – Computes the MSE error of two time series.

- `mape(y_test, y_forecast)` – Computes the MAPE error of two time series.

- `smape(y_test, y_forecast)` – Computes the SMAPE error of two time series.

4. MACHINE LEARNING PIPELINES

Assembly lines were invented in the early 1900s to reduce time and cost in industrial production. Pipelines are the software counterpart to assembly lines. Software processes are a sequence of well-defined steps amenable to automation. Sklearn and TensorFlow, among other libraries, implement pipelines, providing different functionalities.

A pipeline aims to assemble functions whose output is fed as input into the following pipeline step. Pipelines, thus, implement a form of function composition. This function composition automates a machine learning process and reduces the introduction of bugs in its reuse. Fig. 4. shows an example of a machine learning pipeline.

Fig 4. An Example of a Machine Learning Pipeline

At each step of assembling a pipeline, there may be more than one option. For instance, you may decide to use an MLP or the Nearest Neighbor method for forecasting. So, although the pipelines are linear, producing the final version of it is not. It may even involve some backtracking; it usually does.

We need a formalism to explore those possibilities. You will provide this formalism in the form of a library that allows all the required functions to perform this exploratory analysis and the formation of what we will call a Transformation Tree. Fig. 5. shows an example of a Transformation Tree.

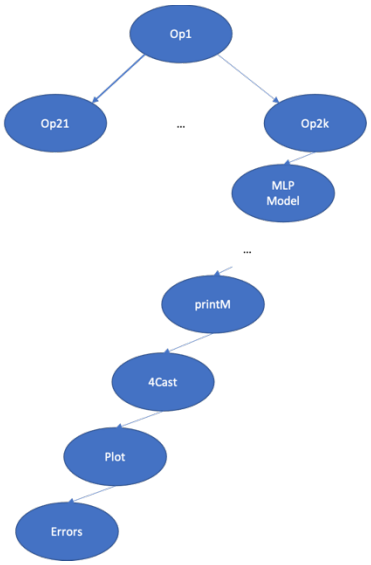

Fig. 5. Tree of Transformation Operators

While testing and debugging the required transformations to produce an acceptable ML model, we may want to exchange one operator for another, replicate complete branches of the tree, add branches or children to a node, etc. The whole tree will be changing with time, along with the experimentations a data scientist performs.

The whole tree represents a set of alternatives to the production of a reasonable forecasting model. At any point in the tree's development, we can execute it, test all different alternatives, and compare their performances. Note that this action may take a very long time.

Once a set of transformations proves to work for the given task at hand, the subtree that represents the most reasonable solution (a path of the tree) can be extracted and converted to a pipeline. Figure 6. shows an example of a pipeline extracted from a Transformation Tree.

Pipelines are fed with data and executed. If you have a set of datasets, you can apply the same pipeline to all its members, easing the process and reducing programming errors.

After producing a forecasting pipeline, it goes into production. During this machine learning product's lifetime, it needs to be monitored, maintained, and updated regularly. The underlying process may change, and the model may no longer be adequate. All those circumstances may lead to more experiments, i.e., to produce changes to the transformation tree.

Last but not least, experiment tracking is essential. A Transformation Tree is a mechanism that allows a data scientist to keep track of the performed experiments. The lack of an experiment log may take you to walk in circles.

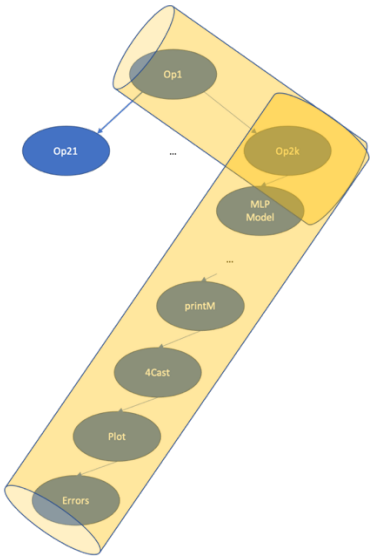

Fig. 6. A Path on a Transformation Tree Converted to a Pipeline

5. TRANSFORMATION TREE LIBRARY

This project aims to develop a library for the manipulation of Transformation Trees. This library must include the following functionalities:

i. Create a new tree.
ii. Add operators to the transformation tree, checking type compatibility of output/input of operators.
iii. Replace a process step with a different operator.
iv. Replicate a subtree.
v. Replicate a tree path.
vi. Add a subtree to a node.
vii. Load/Save a tree.
viii. Load/Save a pipeline.
ix. Execute a tree.
x. Execute a pipeline.

The implementation details of those operations depend heavily on the design decisions of the tree data structure. Note that Transformation Trees are *n*-ary trees.

6. TECHNICAL REQUIREMENTS

This section presents a minimum set of technical requirements. Your projects should generate many more than just these.

i. The delivered system must be complete. Even if you deliver a subset of the required functionality, the provided system must execute without any major bugs. The graders do not have time to debug your software.
ii. Systems should use standard libraries to the extent possible. Follow the guidelines provided to document your code and mention what each library provides.

iii. Installing and running the system. Provide the required instructions to install, execute, and use your system.
iv. If you are programming in Python, use virtual environments (pip or conda) and include a requirements.txt file in your submission so that graders can reproduce the exact conditions under which you developed the library.
v. All names you use for files, variables, functions, libraries, etc., must comply with the python standards [8]. Please, keep the function names exactly as they appear in this document (i.e., those in `monospaced font`); you are free to choose any new function names (e.g., those in Section 5.)

7. TEAMS, MEETINGS, AND DELIVERY

You will be working on groups of five students. Early in the class, we will study team organization; you will use those concepts to organize your team, divide work, etc. Before each scheduled meeting with the customer (see below), every student will turn in a peer evaluation form. Keep that in mind when accepting your part of the work. Also, each team must submit a brief report of the team organization. More about this will be detailed in class.

In a real-life project, you need to meet with your customer or a designated committee for the following reasons, among others:

i. Initial meeting – that is where the customer specifies the need for a new system.
ii. Questions and concerns – developers may have questions about the processes to be included in the system or the system environment (its ecosystem).
iii. Quality specifications – these are specifications required by the customer or proposed by the developer.
iv. Specifications of standards – standards represent general knowledge of the application field.
v. Changes to specifications – procured by the customer. These better happen before you start any implementation or detailed design. You must sign a contract before the project begins.
vi. Progress reports – either previously agreed and scheduled, requested by the customer, or proposed by the developer team.

Taking this scenario to this course project:

- I will be your customer.
- Each team will be assigned two meetings before the final presentation.

Project delivery is the final presentation of the project to the customer. For larger projects, there may be several deliveries at the end of each planned project stage.

Preliminary meetings will occur during my office hours, and the final delivery will take place during class. Those meetings appear in the class schedule, but their dates may change slightly, depending on your advance. In this project, the final delivery will occur during class and in the form of a brief presentation.

8. REFERENCES

[1] Montgomery, Douglas C., Cheryl L. Jennings, and Murat Kulahci. Introduction to time series analysis and forecasting. John Wiley & Sons, 2015.
[2] Box, George EP, et al. Time series analysis: forecasting and control. John Wiley & Sons, 2015.

[3] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.
[4] Nielsen, Aileen. Practical Time Series Analysis: Prediction with Statistics and Machine Learning. "O'Reilly Media, Inc.," 2019.
[5] Bengio, Yoshua, Ian Goodfellow, and Aaron Courville. Deep learning. Vol. 1. Massachusetts, USA:: MIT Press, 2017.
[6] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *the Journal of Machine Learning Research* 12 (2011): 2825-2830. (https://scikit-learn.org/)
[7] McKinney, W., et al. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).
[8] Pandas development Team. Pandas-dev/pandas: Pandas. *Zenodo* 21 (2020): 1-9. (https://pandas.pydata.org/)
[9] Kantz, Holger, and Thomas Schreiber. *Non-linear time series analysis*. Vol. 7. Cambridge university press, 2004.
[10] Guido van Rossum, Barry Warsaw, Nick Coghlan. PEP 8 -- Style Guide for Python Code. Python Software Foundation. Aug. 2013. (https://www.python.org/dev/peps/pep-0008/)