Time series forecasting github repo: Leveraging Francesca Larezzi’s materials

This document provides an assessment of the slides and notebooks created by Francesca Lazzeri on Time Series Forecasting. It attempts to determine whether this content should be included in a Forecasting (or another name) Github repository, and if so, what should be done to make it fit that format.

The materials reviewed are:

* Set of notebooks and slides compressed in a [zipped file](https://microsoft.sharepoint.com/teams/Vienna/Shared%20Documents/Recipes/forecasting-frlazzeri.zip)
* Recent [AML deployment notebooks](https://notebooks.azure.com/frlazzeri/projects/AMLsTimeSeriesDeployment)

The section below provides a summary of this evaluation. Details are available from the second page.

# Summary

Overall observations:

* Nice notebooks and slides, as they take the user through important steps of time series forecasting
* Information on algorithms will need to be re-organized, so concepts are defined before they get used
* There should likely be several sections -- For instance:
  + Forecasting fundamental concepts -- could be a readme/set of informational pages
  + Data pre-processing/Feature engineering\*
  + Modeling methods\*:
    - Traditional statistical algorithms
    - Deep learning solutions
      * This section should also contain information about when to use a particular algorithm and when not to, or when to use a method and not another one
      * Some of the notebooks from Modules 1 and 3 should be moved here
  + References, i.e. links to:
    - Francesca's work
    - Azure's documentation
    - Keras' documentation
* In the notebooks:
  + More explanations are needed around:
    - Algorithms used -- even if the theory was described in the Readme.md or other informational page in the repo, a summary or reference to that informational page needs to be provided in each notebook
    - Results obtained and how to interpret them
  + Code needs to be tidied up (there are a lot of repetitions) and written in a more pythonic way
* Some of the content (slides and notebooks) don't need to be moved over to the Forecasting repo (cf. details below)
* Notebooks that leverage AutoML will need to be discussed to see if they should be replaced by computations on AMLCompute

*\*could be in their own folders*

# Assessment details

## README.md

The tutorials leverage the Azure Notebooks platform and free compute

* Notebooks would become accessible from the Forecasting repo
* Users would leverage:
  + Their workspaces
* AMLcompute or other compute resources, and not necessarily the free compute option

## Config.json

* Contains Microsoft's internal subscription id, resource group and workspace name
* Would not be available in the repo, but would use the usual Workspace methods to get at the user's information for them to run the notebooks in their own workspaces

## Library.json

* Contains info on NYC conference
* Not needed for our purpose

## Slides

* #2: Indicates who the content is addressed to and what the pre-requisites are
* Could be moved to a readme.md file on the github repo
* #5 and structure of the folders provided could help with the structure of the repo, within the repo's "examples/" folder
* We may want to replace the content developed with AutoML by other AML solutions (it may not be obvious, though – cf. note notes on notebook below)

### Module 1: Introduction to Time Series Forecasting

* + Slides 8 to 21 + 29-31: Fundamental concepts >> These should be part of a Readme or separate informational document in the repo, which explains the following topics:
    - Why to study time series
    - What is a time series, and how does it different from other typical ML datasets
      * What are the major components of a time series: level, trend, seasonality, noise + show example plots
      * What is stationarity
      * How to make a time series stationary --> this could be notebook on its own
    - What is time series forecasting and how does it differ from time series analysis
    - What pieces of information matter when doing forecasting:
      * Time horizon: what is it?, what types exist?: short vs medium vs long term
      * Temporal frequency of the data
      * How often to run the prediction?
      * How often to retrain the model?
      * How much data are needed?
    - What does "grain" mean? (cf. notebook on forecasting of orange juice sales - Module 2)
    - What is the forecast horizon? How does that relate to short/medium/long term?
  + Slides # 22 + 36-47 + 49-58: Modeling:
    - Mathematical models (with formulas and explanations, and when to use them and when not to):
      * Additive:
        + Simple exponential smoothing
        + Exponential smoothing
        + ARIMA
      * Multiplicative
      * Mixed
      * deep learning
        + [This blog post](https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/) also provides a list of methods (which somewhat overlaps with this one) and explains what they are about
  + Slides #23 - 26: Feature engineering
    - Lag for uni- and multi-variate datasets
    - Multi-step lag
    - Window features (summary of values over fixed window of prior time steps)
    - This could be a notebook of its own
    - Info would be needed to help users figure out how to handle categorical variables (e.g. is business hour or not)
  + Slide #32: Typical forecasting model creation workflow

### Module 2: Introduction to Azure ML Service (slides 64 - 86)

* + Overview of Azure product line --> is this something we want to have in the repo?
    - If so, how much of this do we want?
    - Should we instead reference [Jingyan's notebook](https://github.com/Microsoft/Recommenders/blob/master/notebooks/run_notebook_on_azureml.ipynb) on how to run a notebook on AMLCompute?
  + Data prep (slides 88 - 97):
    - Slides cover generic data prep steps --> it may be better to discuss what steps are typically taken for time series data:
      * E.g. date formatting, data augmentation (e.g. holidays vs not, business hours vs not, time of day, day of week, etc.), interpolation (to get data points at regular intervals), etc.
  + AutoML (slides 98 to 107):
    - Introduction to AutoML --> Content likely not needed in the forecasting repo -- If we want it, though, we could just point to [already existing documentation](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-configure-auto-train)
  + Tool agnostic python SDK (slides 107 - 109):
    - Not needed in the repo

### Module 3: Introduction to RNN for time series forecasting (slides 111 - 142)

* + 111-127: Introduction to NN concepts (weights and bias, activation functions, forward- and backward-propagation, loss function and cost, batch/stochastic/mini-batch gradient descent)
    - This could be part of a general Readme or other informational document in the repo, or links to similar content could be used
  + 129-142: Introduction to RNNs
    - Introduces to RNN concepts (backpropagation through time, Vanilla, LSTM, GRU, exploding and vanishing gradients and how to solve such problems)
      * This should also go in a redme/informational page
    - Slide 132: too simplified (3 sets of weights and biases) --> better explained in slide 134

### Module 4: Build your own time series forecasting model

* + Slide 144: Short vs long term prediction:
    - This table could go in one of the Readme/informational pages. It provides clues on:
      * what short, medium and long term predictions mean (<2 days, 2 days - 1 month, > 1 month)
      * What data granularity can be (hourly vs daily frequency)
      * Use cases
      * Features that can be relevant to the prediction
      * The amount of data for the model to be robust
      * Success metrics
      * How often to generate predict and/or retrain the model
  + Slide 151, 152 and 155 - Show:
    - How the data need to be split, i.e. sequentially (and not randomly as in most ML problems)
    - 2 of the core principles of ML model training:
      * Train with training set, fine tune/choose best model with validation, and evaluate with test set
      * Normalization/scaling
    - Do we want these points to be made in the Readme/informational pages? If so, we may need a general "ML core principles" section
  + Slide 158 = 158 repeat of slide 60
  + Slide 159: Keras
  + Slides 161 - 187:
    - Cover some of the content explained in the notebooks
    - Having this whole content in a readme/informational page would be helpful, as it would provide a clean version of the process, before users actually work on it in the notebooks
    - Slide 176: It would be good to explain what min\_delta and patience represent, and why early stopping stops where it does
    - Add explanations to content of slides 185-187
  + Slides 189 - 197:
    - Not needed in the repo

## Notebooks

* Link to datamarket website is broken -- need to find a better one

### Module 1

#### 1 - Load and handle time series in pandas

* Quite repetitive
* Functions/methods used not necessarily timeseries-specific
  + This notebook doesn't need to exist by itself. Its content can be integrated into others

#### 2 - Feature engineering for time series

* + Helpful notebook
  + Improve way to construct the initial dataframe

import pandas as pd

df = pd.read\_csv(data\_path, header=0, parse\_dates=True, index\_col=0, names=['temperature'])

df['month'] = df.index.month

df['day'] = df.index.day

* In this notebook too, a lot of the commands are repeated -- I understand that this is to allow users to select the cell they want and run it with their own data, but:
  + The repeated commands should be removed
  + More explanations are needed, especially for the window and rolling features
  + Links to python libraries or to small scripts are needed (when they exist) for the list of possible features that can be created:
    - E.g. business hours --> could be a script with 3 inputs: start and end of business day + initial dataframe --> returns dataframe with extra categorical column
    - E.g. Python [holidays](https://pypi.org/project/holidays/) library
  + Examples should be provided to show how to:
    - Determine time series frequency (hourly, daily, weekly, etc.)
    - Impute missing data
    - Encode categorical variables
      * (AutoML does that automatically, according to the auto-ml-forecasting-orange-juice-sales.ipynb notebook from Module 2)

#### 3 - Stationarity of a time series

* + Overall, interesting notebook, but not enough information on what stationarity is and how to interpret the results from the different tests that would allow the user to decide whether a time series is stationary or not
  + We may want to start by showing the different components of a time series (even if we may already have such a graph in the Readme file we will create) -- this way, everyone understands what component to look at to determine the stationarity status of the time series
  + Comments are needed, especially in the "Summary statistics to check stationarity …"
    - Plot and results don't help understand whether the time series is stationary or not
  + Need an explanation of what is returned by the Dickey-Fuller test and how to interpret the results
    - What is the null hypothesis here?
      * Is it that the time series is constant? If so, the results seem to suggest that we can reject that null hypothesis, but the plot of the time series doesn't to show much of a trend
        + Answer available [here](https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/)
  + Same test on Airline data --> no need to copy the whole cell again --> just have the 2 "read\_csv" command in same cell, one with birth data and one with the airline ones, commented out
  + Add link to wikipedia or other reference on Dickey-Fuller test
  + Add instructions on how to make a non-stationary time series stationary (e.g. of Airlines dataset) -- this is currently missing from the notebook

#### 4 - Auto-regression model

* Needs explanations on:
  + What a **static** autoregressive model is and how that differs from a **dynamic/updated** one
  + why train/test split is done the way it is
  + What the coefficients represent --> seem to be coeffs of the different lagged values + a constant
  + What the "walk forward over time steps in test" section means and does, and why would the results be better than way than when using a static model?
  + Do we want to discuss prediction interval?

#### 5 - Moving average (MA)

* Need to explain:
  + why considers t as the prediction, and doesn't do a fit instead
  + what lag = 15 means and how that was computed
  + Why history stops at end of training series - lag
  + Why time series of "t" is the predicted time series -- why is it not time series of "t+1"?
  + Why this notebook is called "moving average" -- it seems to be more about residuals than about a moving average
* Need to plot the initial time series, the one shifted by one time step and the residual time series to actually see whether there is a trend that still needs to be modeled or not
* Cell #5: no need to repeat the entirety of cell #3 here

#### 6 - Autoregressive integrated moving average (ARIMA)

* Assumptions made on the time series when using ARIMA need to be articulated and solutions need to be provided when the dataset doesn't respect obey these assumptions ([Blog post on how to determine whether a time series is stationary or not, and on how to make it stationary in the latter case](https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/))
* Need interpretation of the autocorrelation plot
  + Is the lag 1 or 2 or else time units?
  + How will we use it?
* Explain:
  + Why we use order(5,1,0) --> what does each number represent?
    - Some of this is somewhat answered in slide #57, but not really, because we still don't know what "xth order moving average means" --> actually answered at slide #61
  + Why we use disp=0
  + What each metric returned in the table represents, and what their values mean
  + Whether components with a p > 0.05 is significant and should be kept in the model or not
  + Why some coefficients are complex
  + What the residuals represent
    - Are they the difference between the predictions and the future data (i.e. test set), or something else?
    - Are we satisfied with the distribution of the residuals? Is the distribution not too wide and the tails too long?
  + Why we use ".forecast()" here and not in other notebooks
  + What ".forecast()" returns, since yhat is just output[0]
* This notebook is more typical to other ML workflows, with a training on the training set and a comparison of the results of the forecast to the reference future data (i.e. test set)
* Need comments on the results obtained
  + E.g. is an RMSE of 83.417 a lot or not much in comparison to the time series values?
* It may be good to plot the residuals once again at the end, once the model has been improved

### Module 2

* Readme file refers to image classification notebooks --> We should remove those from there
* Config.json and library.json files can be removed

#### 1 - Run experiment

* Introduces to the concepts of workspace, experiment and run --> would not use this notebook but instead discuss these concepts in a AML-specific notebook that trains a forecasting model on AMLCompute, for instance

#### 2 - Deploy web service

* Same comment as above, so the web service serves a forecasting model, not a generic one
  + Currently investigating ways to create a generic notebook that would leverage the 3 deployment notebooks from CV (on [ACI](https://github.com/Microsoft/ComputerVision/blob/staging/image_classification/notebooks/21_deployment_on_azure_container_instances.ipynb), on [AKS](https://github.com/Microsoft/ComputerVision/blob/staging/image_classification/notebooks/22_deployment_on_azure_kubernetes_service.ipynb), [testing of web service](https://github.com/Microsoft/ComputerVision/blob/ateste-deploy-3/image_classification/notebooks/23_web_service_testing.ipynb)), and could be reused in other repos
* The example provided doesn't really use a model, but just a value, and the service returns the result of a simple multiplication --> it may be better to register and serve a forecasting model instead

#### 5 min data prep intro

* Introduces to DataPrep functionalities -- These could potentially be shown in a notebook that gets data from a blob storage or other resource and builds a time series forecasting model.
* Best would be to use some of these functionalities, and to refer the user to the ["data prep" notebooks](https://github.com/Azure/MachineLearningNotebooks/tree/master/how-to-use-azureml/work-with-data/dataprep/how-to-guides) that Roope just created, in case they want to do things with their data that are not covered by the time series ones
  + In any case, the time series notebooks should only focus on forecasting, and not so much on explaining all the details of the Data prep features

#### Energy forecasting (auto-ml-forecasting-energy-demand-end2end)

* This notebook leverages AutoML, runs potentially up to 11 models (according to [this table](https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-configure-auto-train#select-your-experiment-type)), and finds the best parameters
  + Do we want to replace that by computations on AMLCompute (in parallel?)?
  + The results are not so great --> is it worth showing this autoML example? Would it make sense to show an example of HyperDrive instead, with a well defined algorithm (instead of running through many, as autoML does)?
  + It may be possible to simplify the creation of the Docker image (not sure we need to add "apt-get install -y build-essential gcc g++ python-dev unixodbc unixodbc-dev"
  + If we keep the web service deployment, we need to:
    - Show a way of testing it (service.run() and requests.post())
    - Add commands to delete the ACI resources and the Docker image
  + Should we consider having the deployment in a separate notebook, as we did in the CV repo?

#### Auto ml forecasting orange juice sales

* + Introduces the concept of grain --> should be part of a readme/informational page
  + Same type of operations as in the energy demand forecasting notebook
  + It may be useful to describe briefly what gets collected in the "debug\_log" file

### Module 3: Introduction to NNs for time series forecasting

#### 0 - Data setup

* Doesn't need to be a separate notebook - it actually shows how to retrieve data and how to plot them

#### 1 - time series ARIMA

* The content of this notebook should be moved to the "module 1" section, alongside notebook "6 - Autoregressive integrated moving average" -- these 2 notebooks could actually become only one -- "1 - time series ARIMA" contains some explanations, especially on train/test split and scaling/normalization, that are helpful to the user
* Needs better explanations of p, d, q -- Currently doesn't help the user understand what these parameters do/how they affect the original time series
  + Section #5 of [this blog](https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/) explains better
* Needs explanation/guidance on how to select the horizon value
* Needs explanations on what the parameters to "order" mean and how they were chosen
* Needs interpretation of the results

#### 2 - one step FF univariate

* This notebook is good for an introduction to NN for forecasting
* Needs explanations:
  + On the architecture chosen
  + On how it is possible that the validation error is as good/better than the training one

#### 3 - one step RNN univariate

* Operations very similar to "2 - one step FF univariate" - Only differences are the model used and the shape of the input features
  + As in the previous notebook, needs explanations on:
    - Choice of the architecture (maybe chosen for simplicity)
    - Why validation error is better than training one
  + It would be interesting to display the results of the 2 methods in the same plot (CNN seems to perform better than RNN in this case)

#### 4 - multi-step RNN vector output

* This notebook is the continuation (complexification) of the prior one (i.e. the mechanics are the same)
* It shows a model with:
  + 2 independent variables, instead of one
  + The prediction over the next 3 time steps, instead of over the next one
* This notebook could be kept close to as is, as, by this time, the user knows what to expect/understand what is going on
* A plot comparing the results at t+1 from this model to those obtained from the previous model may help see the benefits of having > 1 independent variable in the model

#### 5 multi-step RNN encoder-decoder

* This notebook is the continuation (complexification) of the prior one
* Here, new concepts, model layer types and parameters are introduced:
  + Encoder, decoder
  + RepeatVector, TimeDistributed
  + return\_sequences=True
    - All of these should be explained (i.e. what the do and why we use them)
      * E.g. "RepeatVector(HORIZON)" repeats the output (He6) of the encoder layer 3 times (HORIZON = 3 here), i.e. as many times as the horizon chosen
      * "TimeDistributed(Dense(1))": Applies the "dense(1)" transformation to each of the 3 output time steps, i.e. converts elements of dimension 5 (LATENT\_DIM) into dimension 1 (i.e. a scalar)
* Here too, a comparison to the results of the simpler model (i.e. from multi-dtep RNN vector output) would be interesting to see -- An interpretation would need to be added too

#### Quiz notebooks

* Can be ignored, as they cover the same topics as the others, and are there to check that the workshop participants understood the concepts and could figure out what Keras layers to use on their own

#### Run notebook in AML

* Could be replaced by the content of [Jingyan's notebook](https://github.com/Microsoft/Recommenders/blob/master/notebooks/run_notebook_on_azureml.ipynb), which leverages the latest version of the SDK functions developed for the purpose of running a notebook on AMLCompute using papermill
  + Jingyan's notebook would need to be adapted with:
    - 1\_time\_series\_arima.ipynb as the referenced notebook
    - The proper set of python libraries (scikit-learn, pandas, statsmodels, matplotlib, etc.)

#### Reference Notebooks

* This is a set of notebooks from which the other notebooks were derived
* They can be leveraged to extract some extra information/interpretations

##### One\_step\_RNN\_multivariate.ipynb

* Does not have an equivalent in the main folder
* It shows the prediction of the next time step from multiple independent variables
* It is similar to "3 - one step RNN univariate" but with 2 variables: load an temperature

##### Multivariate\_FF\_vector\_output.ipynb

* Is the continuation of "2. one\_step\_FF\_univariate.ipynb"
* It shows how to predict over several time steps, from more than 1 independent variable, using a CNN
* It would need to be fixed, if we were to use it, as it seems to be currently broken

##### Multi\_step\_RNN\_encoder\_decoder\_teacher\_forcing.ipynb

* It is a more advanced notebook, where the prediction output of one step is fed to the next time step prediction

Those 3 notebooks could be added to the repo

### Module 4: Build your own time series forecasting model (stock exchange and trading data forecasting)

#### RNN\_LSTM\_GRU.ipynb

* This notebook is actually the example run with the stock exchange data
* The quality and positioning of this notebook are not as good as in the others
  + Scaling data section: The comment about mean =0 and variance = 1 is true if we use the StandardScaler() object from scikit-learn, not the MinMaxScaler(), as done here --> either change the comment or change the scaler used
  + Text needs proof-reading
  + Dataset needs to be described and plotted before normalizing it
  + Needs explanations on why the dataset is not split as before, and why we consider sequences of 20 time steps here --> the way it is done is such that the sequences are 19 values long, not 20
  + The explanation provided after cell #4 is misleading, it should say something along the lines of:
    - First 4497 points represent the training set
    - Following 562 points correspond to the validation set
    - The last 562 points constitute the test set
      * Each point is composed of 4 values: one for the "Open", one for the "High", one for the "Low" and one for the "Close" time series
      * Each point is a sequence of 19 values, i.e. the value at the current time stamp along with the 18 timestamps that preceded it
  + The graph is also weird -- how come x\_train and y\_train don't have the same number of points? Same for x\_ and y\_valid, and x\_ and y\_test.
  + Needs explanations around:
    - The parameter values chosen
    - The architecture used
  + A visual representation of the model architecture would be helpful
  + This notebook uses Tensorflow and not Keras --> a lot could likely be removed/simplified if we were to use Keras instead (e.g. shuffling and generation of batches, cost function, optimizer)
  + The display of "4.99, 9.98, 14. 98, etc. epochs looks weird" --> Using Keras directly would return the info we need and in the proper format
  + It would be great to show the training and validation error --> Using Tensorboard would be helpful here
  + Needs explanations on:
    - Why predictions get further away from actual values, as time goes by
    - What peepholes connections are and why we should use them
  + Show the actual results obtained with basic RNN cells, LSTM cells and GRU cells + comment on which one is best (in terms of MSE, speed, etc.)
  + I would recommend not including this notebook to the repo, as it doesn't bring more info than already covered by the other notebooks

#### Stock Market \Predictions with LSTM in Python.ipynb

* + This notebook is much better than the previous one in terms of quality of the text and progression
  + It introduces to the concept of window normalization
    - In cell #9 (which failed):
      * The same scaler is fitted over successive time windows -- I am not sure this is the best way to proceed -- I would think that each window must be normalized independently, and not benefit from the normalization of the prior time windows -- If we keep this notebook, this is something we may need to investigate/modify
      * The code leading to the error message shown should be fixed
        + The person who wrote the notebook assumed there were > 10k data points, but there are only 3393, hence the error
        + Replaced content of that cell and following ones by:

nbr\_points = len(mid\_prices)

percent\_split = 0.1

train\_end = int((1-percent\_split)\*nbr\_points)

train\_data = mid\_prices[:train\_end]

test\_data = mid\_prices[train\_end:]

scaler = MinMaxScaler()

train\_data = train\_data.reshape(-1,1)

test\_data = test\_data.reshape(-1,1)

# Train the Scaler with training data and smooth data

smoothing\_window\_size = 300

for di in range(0,train\_end,smoothing\_window\_size):

print(di, di+smoothing\_window\_size)

scaler.fit(train\_data[di:di+smoothing\_window\_size)

train\_data[di:di+smoothing\_window\_size] = scaler.transform(train\_data[di:di+smoothing\_window\_size])

# Reshape both train and test data

train\_data = train\_data.reshape(-1)

# Normalize test data

test\_data = scaler.transform(test\_data).reshape(-1)

# Now perform exponential moving average smoothing

# So the data will have a smoother curve than the original ragged data

EMA = 0.0

gamma = 0.1

for ti in range(train\_end):

EMA = gamma\*train\_data[ti] + (1-gamma)\*EMA

train\_data[ti] = EMA

# Used for visualization and test purposes

all\_mid\_data = np.concatenate([train\_data,test\_data],axis=0)

* The code could be simplified and leverage the fit\_transform() method instead of fit() and transform(), separately
* Cell #10 assumes that the scaler used on the test data is the one used for the transformation of the last window of training data
* Cell #11: Explanation needed around exponentially moving average -- or at least a reference to a document that discusses it
* The explanation of EMA is backward: y = 0.1 means that the current value of x contributes at 90%, according to the formula --> For the text to be correct, y = 0.9 should be used instead
* running\_mean = running\_mean\*decay + (1.0-decay)\*train\_data[pred\_idx-1] should be running\_mean = running\_mean\*decay + (1.0-decay)\*train\_data[pred\_idx] -- not sure why pred\_idx-1 is used, instead of pred\_idx
  + According to the formula, we should use:
    - running\_mean = running\_mean\*decay + (1.0-decay)\*train\_data[pred\_idx]
    - And plt.plot(range(1,N+1),run\_avg\_predictions,color='orange', label='Prediction')
* Last 2 sentences of "Predict more than one step into the future" are repeated
* LSTM is not a model but a cell type in a recurrent neural network
* Showing the formulas of the LSTM gates, in addition to the image, would be helpful or a link to a document explaining it
* The DataGeneratorSeq class needs a docstring and explanatory comments
* As in the previous notebook, Tensforflow should be replaced by Keras (easier to deal with and to understand)
* The explanations of RNN provided in the notebooks of module 3 are a lot better than in here
* A diagram of the model architecture would be helpful
* Not sure when the notebook was created, but the Adam optimizer is not "very recent"
* The "Running the LSTM" section would need to be fixed because of:
  + array length issues

if w\_i + pred\_i < len(mid\_prices): <-- add that if statement

mse\_test\_loss += 0.5 \* (pred - all\_mid\_data[w\_i + pred\_i]) \*\* 2

* Code indentation problems
* "Visualizing the predictions"
  + Change suplotting with:
    - plt.subplot(3,1,1), plt.subplot(3,1,2) and plt.subplot(3,1,3)
    - plt.xlim(11000, 12500) should be removed on both the 2nd and 3rd subplots --> replace them by plt.xlim(train\_end,nbr\_points)
  + The true values and predictions need to be converted back to the proper range
* I would recommend not including this notebook to the repo - it is complicated and most of its content is discussed in other notebooks

#### 1\_timeseries\_training\_AML.ipynb

* This notebook shows how to send a training script to AML and run it there
  + We could use it, but it would need more explanation than currently available
  + We could, alternatively, reference [Jingyan's notebook on how to do remote training](https://github.com/Microsoft/Recommenders/blob/master/notebooks/00_quick_start/sar_movielens_with_azureml.ipynb), which contains a lot more explanations on the Azure components involved

#### 2\_timeseries\_deployment\_AMLs.ipynb

* This notebook shows how to deploy a trained model to ACI
  + We could use this notebook, it would need more explanations
  + We could alternatively leverage the notebook published in the CV repo on [deployment on ACI](https://github.com/Microsoft/ComputerVision/blob/staging/classification/notebooks/21_deployment_on_azure_container_instances.ipynb) and on how to test a web service from within and outside the workspace­­­­