Title: Decline Curve Analysis Code Challenge v1.0

1. Abstract/Description

1.1. High Level Overview

Decline curve analysis is a non-trivial task which requires a lot of statistics and reliable algorithms to **precisely** make forecasts. Moreover, outliers in data make it more complicated to forecast accurately. It is suggested to use a machine learning outlier detection algorithm such as Isolation forest: https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e

It is a well-established algorithm in commercial applications. Moreover, this method has a low linear time complexity and a small memory requirement. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of the size of a data set. https://blog.easysol.net/using-isolation-forests-anamoly-detection/

The provided data gives a good starting point on forecasting Gross_oil_bbls and Gross_gas_mcf. So, we will address the following features:

- Daily_rdg_date
- Gross_oil_bbls
- Gross_gas_mcf
- Well_status_desc
- Well_num

The previous R code uses statistical information such as standard deviation, mean to calculate outliers. But we must manually set a threshold to filter those outliers. This method is inefficient, error-prone, not scalable and not precise. The main idea of proposed outlier analysis solution is to use an unsupervised machine learning (ML) approach to detect outliers. The outliers are not only detected but also classified into data outliers, operational outliers, well downtime outliers.

The forecasting process will be established as follows:

1. Prepare, preprocess and clean input data.
2. Identify, analyze, classify and remove outliers from data.
3. Identify initial time to peak.
4. Find the most recent decline by performing peak analysis.
5. Split the data at some ratio, say 90% and 10% based on time.
6. Fit 4 decline curves namely exponential, harmonic, hyperbolic and hyperbolic to exponential on 90% of the data. Allow for setting bounds on factors like b, d, beta and EUR.
7. Build an evaluation pipeline to compare all the curve fits.
8. Find the best fit based on RMSE, AIC, BIC scores and slope of the decline.
9. Forecast the data for wells in the test_ops_results_blank.csv file.
10. Visualize the fit and forecasted data for each well's oil and gas declines. Use visualize.py file to do this.

This approach allows to significantly improve the fitting accuracy and incorporate outlier filtering algorithms. This research provides an an end to end decline curve analysis system along with outlier filtering capabilities.

1.2. Algorithm Outline

This approach recommends a customized approach to solve the problem. A decision tree-based approach called Isolation Forest to outlier analysis is suggested. Use of “trf” algorithm to fit different decline curves such as hyperbolic, hyperbolic to exponential, exponential, harmonic is suggested. SciPy’s curve_fit is used to fit the curves. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html. The functions are implemented in python 3.6.

The algorithm works as follows:

1. Data preparation is important. First the training data is loaded and then gas and oil data for each of wells are separated into different panda’s series objects. The target parameter Gross_gas_mcf and Gross_oil_bbls are considered.
2. Basic signal features such as min, max, median, mean, minimum value of the first derivative etc. are identified using NumPy and SciPy library.
3. A total of six outlier filtering algorithms are identified. After a thorough comparison of these algorithms, the best among them, a ML unsupervised tree-based algorithm is selected. The outliers are identified as data(global) outliers, operational(local) outliers and well down time outliers.
    a. Well down time outliers are removed using python’s NumPy library.
    b. Data outliers are removed using Isolation forest algorithm.
    c. Operational outliers are removed using Isolation forest algorithm on a local set of data. i.e. on each decline.
    d. There are also some peaks which do not represent a new decline. These peaks are removed using the width parameter in peak finding algorithm.
4. The preprocessed data is then sent to initial peak estimating algorithm to find the initial decline. Then, it is processed by the peak finding algorithm to identify other declines (if any) in the data.
5. Data is then divided into train and test based on time. 90% and 10%.
6. Each of the declines are fit using all four decline curves using bounds and trf algorithm by SciPy’s curve_fit library.
7. The fitting is then tested on the test data. A time-based array like [1,2,3,4…] is created as a variable t to test the declines on the length of the test data.
8. Forecast the data for a total of 1100 data points. i.e. for three years.
9. Build an evaluation pipeline to compare these curve fits for each well’s oil and well declines. Send the results of the test data and the actual data for that time to get RMSE, AIC and BIC scores. Also, a fourth evaluation metric called the slope of the decline curve is used. It is observed for three wells in the train data (i.e. 877 wells) that the slope of the decline curve is positive. So, to penalize decline curves with positive slopes, this metric is used.
10. A custom algorithm is used to analyze AIC, BIC, RMSE and slopes to get a best model. Although, the algorithm is good, there might be outliers in test data. So, it is recommended to visually check the fit and forecast.
11. To visualize the fit the forecast for all the wells in the test set, a visualization pipeline is set up.
12. The forecasted values and evaluation results are used to fill the test csv file.
13. To optimize for performance and utilize all the cores on the client machine, the code is optimized and written to allow for multiprocessing in python. This greatly reduces the run time of the algorithm. Training and forecasting on 112 wells just take 69-75 seconds on my 2 core 16 GB machine. It is expected to run much faster on the client’s 4 core 16GB machine.
14. Other time series-based forecasting algorithms such as LSTM and ARIMA have been analyzed systematically.

1.3. Algorithm Functional Units

Data Preprocessing

It was done by using python's pandas, NumPy libraries. Also, python 3.6 codes such as prepare_data.py were used. Also, some filtering is done in main.py file.

ML Filtration of Data Outliers

This block is a recommended improvement for the proposed algorithm. It was found that there are outliers in data it affects curve fitting. A thorough research on outlier analysis is performed and six potential algorithms were identified. Then, the data was loaded to the unsupervised Isolation forest machine learning algorithm from the scikit-learn's open source framework: https://scikit-learn.org/ .A Python 3.6 program was provided to do this task: outlier_analysis.py.

Initial time to peak algorithm

A Python 3 program was created to load the wells preprocessed data to find the initial peak i.e. a peak in the first 180 days. Python 3 file used initial_estimate_of_time_to_peak.py.

Peak finding algorithm

To search for the most recent decline, a peak finding algorithm from scipy.signal called find_peaks() is employed. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. This peak finding algorithm improved the overall performance of the system by identifying some operational outlier peaks. Thse peaks have a width close to one that means they do not cause any declines. Peaks with a minimum width of 7 were identified.

Models and curve fitting algorithms

Four curve fitting functions were implemented in python: decline_functions.py. They were then fitted using SciPy's curve_fit function. Bounds for parameters like b, d, beta and EUR were employed. The optimal parameters from the curve_fit function was used to fit and forecast data.

Evaluation Pipeline

An evaluation pipeline is set up to systematically compare different curve fitting algorithms on different scores like RMSE, AIC and BIC. A best model is selected using a customized algorithm from the four curve fits for each well's oil and gas declines.

Visualization

A python library called matplotlib is used to systematically visualize all 224 fits and forecasts on the test data using a python 3 program called visualize.py.

1.4. Summary and Conclusion

This submission converts the R code to python and improves upon the R implementation in many ways by using customized algorithms and machine learning based outlier filtering. Similar packages for hyperbolic, exponential, harmonic and hyperbolic to exponential have been implemented in python. A thorough research in conducted to identify outliers, analyze and classify them. A total of six machine learning based outlier algorithms have been identified.

An evaluation system is built to systematically compare all fits. Several metrics like RMSE were used. A visualization pipeline is also setup to visualize the fit and forecast. Several forecasting algorithms like LSTM, ARIMA were explored and compared. The curve fits look highly promising and EUR estimates are also calculated for each well.

Unit test analysis with a code coverage of 97% is achieved using pytest library. Code is optimized to run faster using pythons multiprocessing library.

A detailed explanation with diagrams and charts is provided in the next section.

2. Details

Warnings: There are some warnings which repeatedly occur during multiprocessing and unit testing.

A warning like this occur in the run of the code. Do not worry about these warnings. They are not related to the code. Instead there is an open issue in the GitHub for this by the python community: https://github.com/scikit-learn/scikit-learn/issues/12327

PendingDeprecationWarning: the matrix subclass is not the recommended way to represent matrices or deal with linear algebra (see https://docs.scipy.org/doc/numpy/user/numpy-for-matlab-users.html). Please adjust your code to use regular ndarray. This warning occurs because of the latest release of numpy on latest python 3 versions . But, it does not impede our code and system.

Another warning like this, you will find is “threshold attribute is deprecated in 0.22” from scikit-learn, the warning generally appears when one imports tree libraries in python 3.6. The latest version of scikit-learn is 0.20.3. These warnings do not affect the code, because there is no threshold object being used in the code currently.

The above warning is to let us know that in future, we can not use that object. But, there is no need to worry as we are not using one right now.

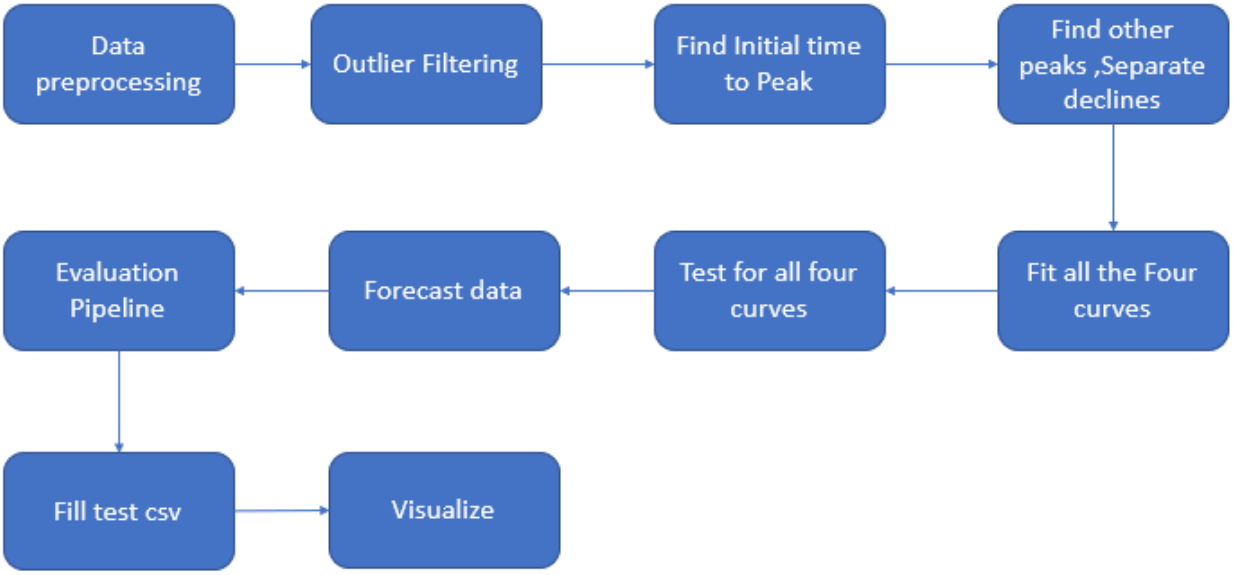


Fig1 : Workflow of the algorithm

2.1. Pre-Processing Input Data

Data preparation:

Provided data is preprocessed using prepare_data.py. The data for oil and gas for each well is separated into different pandas series objects and then stored as a list of pandas series. A pandas series object is like a list, but it has more functions than a list.

Data cleaning:

There are some NaN in the data. Data cleaning is done by using pandas.dropna() function. Also, some wells have preproduction data where all the output from gas, water, oil is zero. This is also removed in my algorithm. This shortens the ramp up period.

2.2 Minimum of data required to comfortably fit a curve?

Data Percentage

It was found that all the wells do not have the same amount of data. So, to comfortably fit a decline curve we need at least 180 data points. This will make sure that the curve is established well and not just a temporary one.

So, minimum requirement: 180 days of production data.

Percentage of data required to train is 90%. 10% can be used to test. So, minimum datapoints required to get a realistic RMSE, AIC, BIC scores are 18.

2.3. ML filtering of outliers

R Code filtration:

There are two types of filtration the R code currently implements :

- Features library outlier analysis.
- Operational outliers using local decline curves.

This process of filtering outliers is very simple, and we also need to set a threshold as how many standard deviations from the mean should we consider a datapoint as outlier? This process is not scalable and not accurate. So, to combat this I have used an unsupervised machine learning algorithm called isolation forest.

Isolation Forest:

The main idea, which is different from other popular outlier detection methods, is that Isolation Forest explicitly identifies anomalies instead of profiling normal data points. Isolation Forest, like any tree ensemble method, is built based on decision trees. In these trees, partitions are created by first randomly selecting a feature and then selecting a random split value between the minimum and maximum value of the selected feature.

In principle, outliers are less frequent than regular observations and are different from them in terms of values (they lie further away from the regular observations in the feature space). That is why by using such random partitioning they should be identified closer to the root of the tree (shorter average path length, i.e., the number of edges an observation must pass in the tree going from the root to the terminal node), with fewer splits necessary.

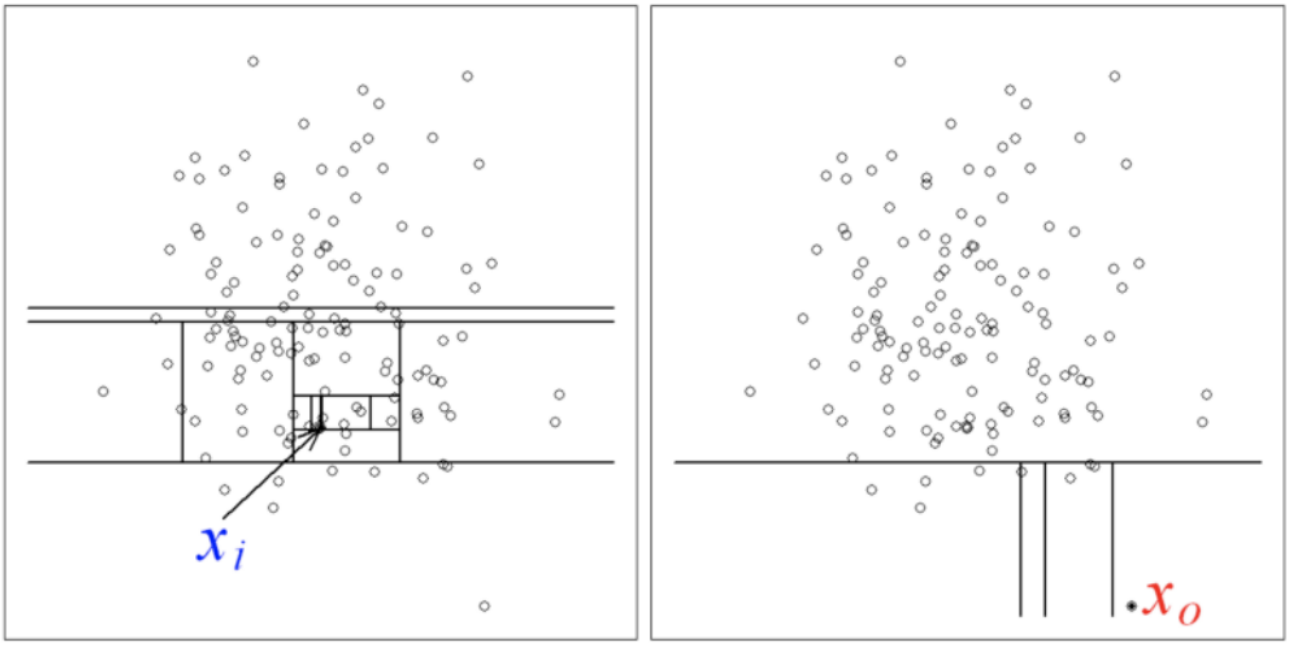


Figure 1 Identifying normal vs. abnormal observations

An anomaly score for each point is calculated and based on the number of points, we calculate anomalies. This advanced approach helps us to improve our model fits.

Literature survey of other outlier analysis algorithms:

1. Z scoring: This method is very simple in that it uses median and standard deviation to calculate a Z score for each point in the data. This is very similar to the one used in R. So, this is not recommended.
2. There are four more algorithms called One class SVM, Local outlier factor, IQR(Inter quartile range) and Random forest filtering.
3. IQR is like Z scoring and One class SVM requires us to scale the data before training the model. This required additional computation and memory. So, among all these algorithms I have chosen, Isolation forest which is both memory efficient, model very well irrespective of the data size.

Exploring the effects of outliers in data:

Outliers in data causes the model fit to be skewed because our model will try to fit that point too causing errors.

1. “Other observations made during the training process of 877 wells is that sometimes an exponential curve fit will have a positive slope because the curve fit also tries to fit the outlier causing an incline rather than a decline.” This is very important because, the curve won’t fit properly.
2. Because of outliers in very rare scenarios, the entire model fits fail. As observed in the case of WellNum-14898_gas. Before the outliers were removed, the curve fits were entirely wrong.
3. But, after the removal, the curve fits appropriately. They are many cases where removing all the four types of outliers improved the accuracy of the fits.

Well Num 14898:

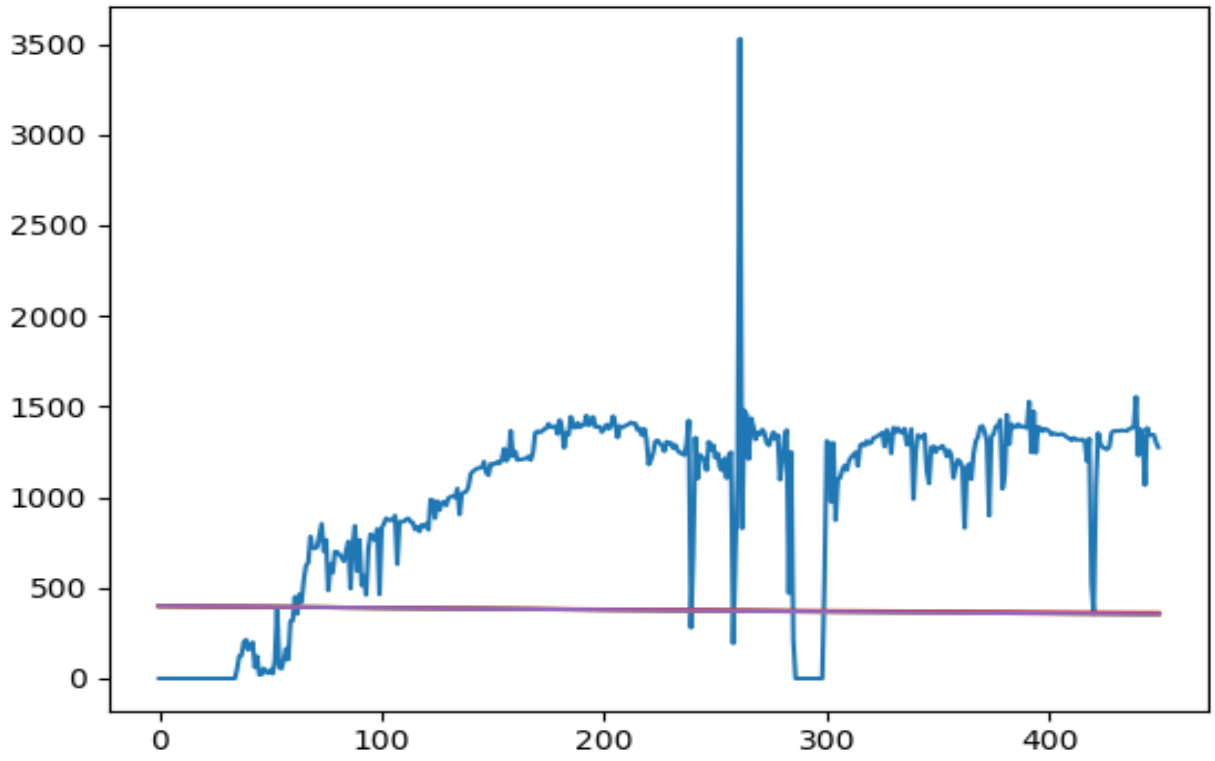


Fig3: Curve fit without removing outliers.

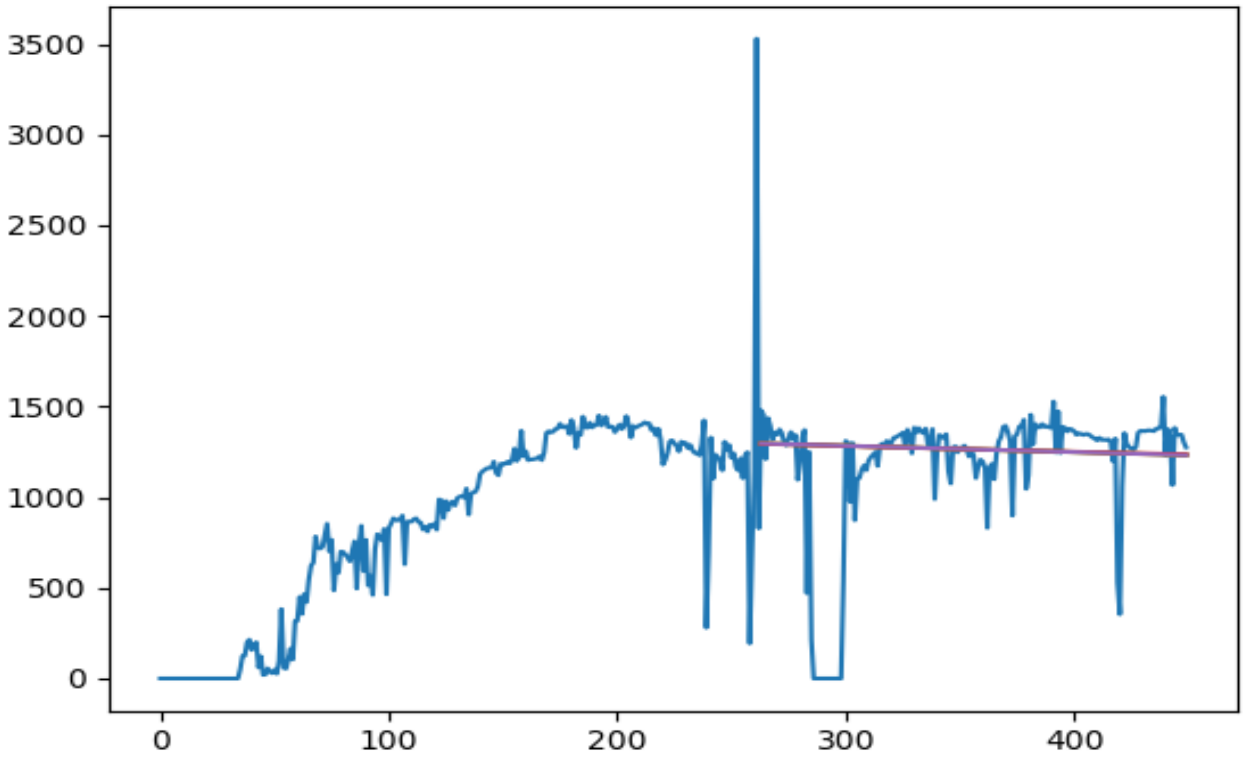


Fig4: Curve fit after removal of outliers

2.4. Initial time to peak and Multiple decline curves

For this research, it is suggested to first find an initial peak in the first 180 points and then find if there are any other declines in the data using peak finding algorithms.

1. Find the initial time to peak and the peak value, peak index in the first 180 points.
2. See initial_estimate_of_time_to_peak.py for a detailed algorithm on how to do this.
3. After that divide the data into two parts. Second part starts from 180 data points after the initial peak. Because we have set a restriction on the minimum horizontal distance between the peaks.
4. Now, we calculate the peaks in seconds part of data using find_peaks from scipy.signal library.
5. Store all the peak indexes and send that to get_seperate_declines function to separate the data into several declines(if any).
6. This separation allows us to find the most recent decline as well as to find multiple declines.

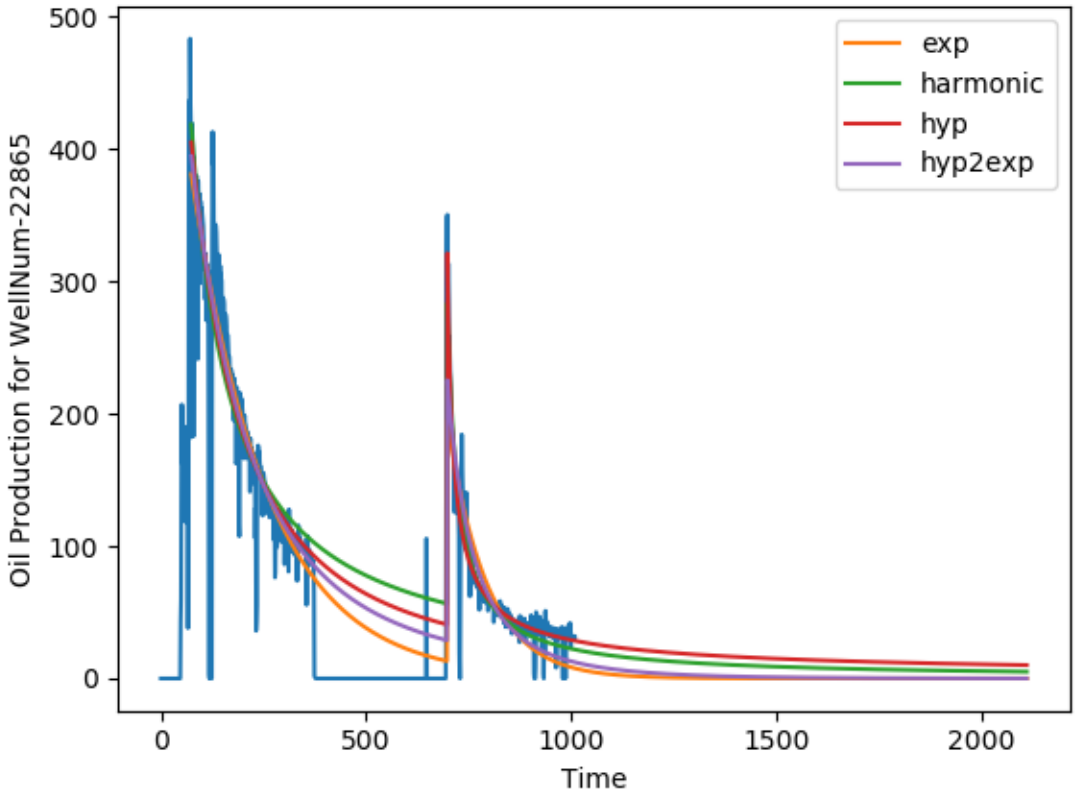


Fig5: Curve fit for multiple declines using the peak finding algorithm.

2.5. Model fitting

The model is fit using the following curve fit functions.

- Exponential: It is observed that this curve fit has a rather fast decline compared to others.
- Harmonic: As the b value is equal to 1, it has relatively less steep curve.
- Hyperbolic: It is observed that the b value for hyperbolic sometimes crosses over 1.
- Hyperbolic2exponential: This is recommended for Oil Curves.

All the curve fits are fit using “trf” algorithm because of bounds. So, this algorithm is perfect. The algorithm fits very well. https://www.sciencedirect.com/science/article/pii/S0307904X16304589. SciPy curve_fit function is used to implement the above trf algorithm . All the functions for decline are written in python itself.

See decline_functions.py for the formulas implemented for each of the above four curves.

2.6. Evaluation Pipeline

There is a total of four metrics used to evaluate the curve fits.

1. RMSE score: It is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the data points are.
2. AIC score is calculated as follows:
3. K = number of variables, here it is one.
4. AIC= 2k - 2log (SSE)
5. Where SSE is the sum of square residuals errors.
6. SSE = (Mean square error)* mean
7. k = number of variables
8. n = number of observations
9. BIC = n*ln (SSE/n) + k*ln(n)\
10. Slope of the decline curve is also calculated.
11. This analysis is done only for the most recent decline in the data.

2.7. Evaluation Results and Analysis report

1. The best model is selected using all the scores. So, for a model to be selected as best , it must have low scores in at least two of the three scoring metrics.

Report:

2. For oil, it was observed that hyperbolic curve fits have less AIC,BIC and RMSE scores second to hyp2exp.

3. For gas, it was observed that, exponential, hyperbolic and harmonic curve fits are commonly identified as best.

4. But, I recommend visualizing another curve fits too. I have provided all the curve fit forecasts in the test csv.

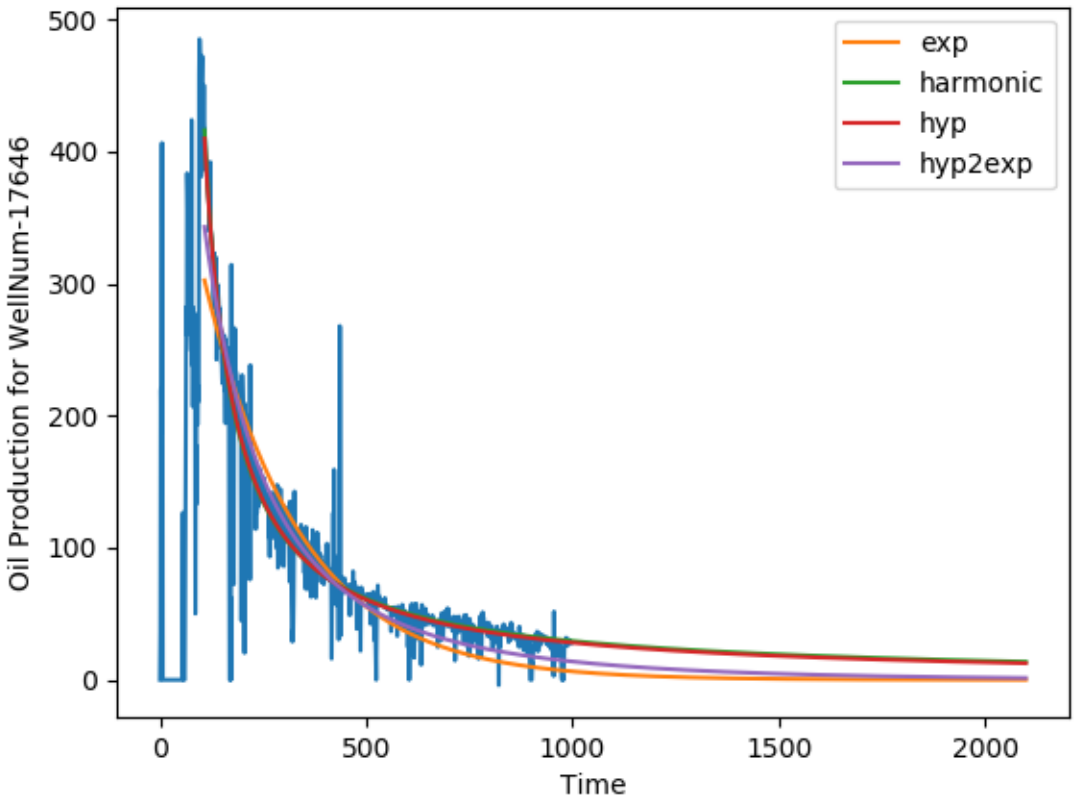
See evaluation_results.csv for all the scores for all wells. It has the best model for each well and the corresponding scores in detail.

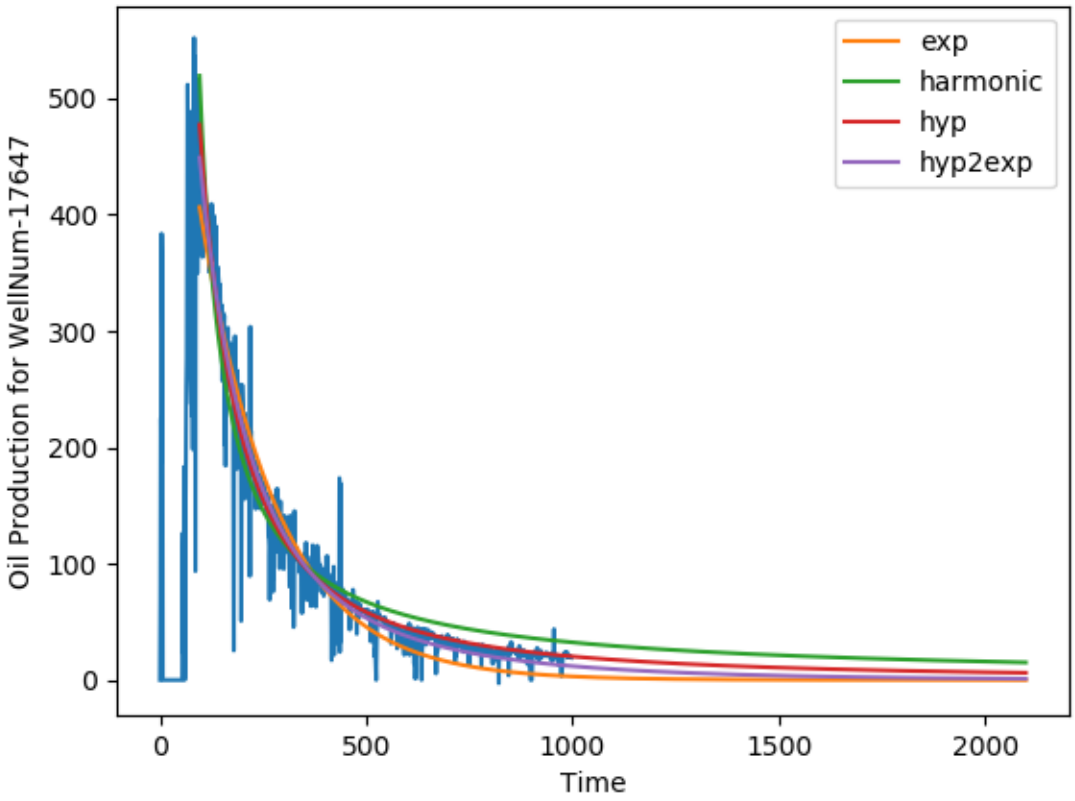
Please check the above csv for a clear analysis report.

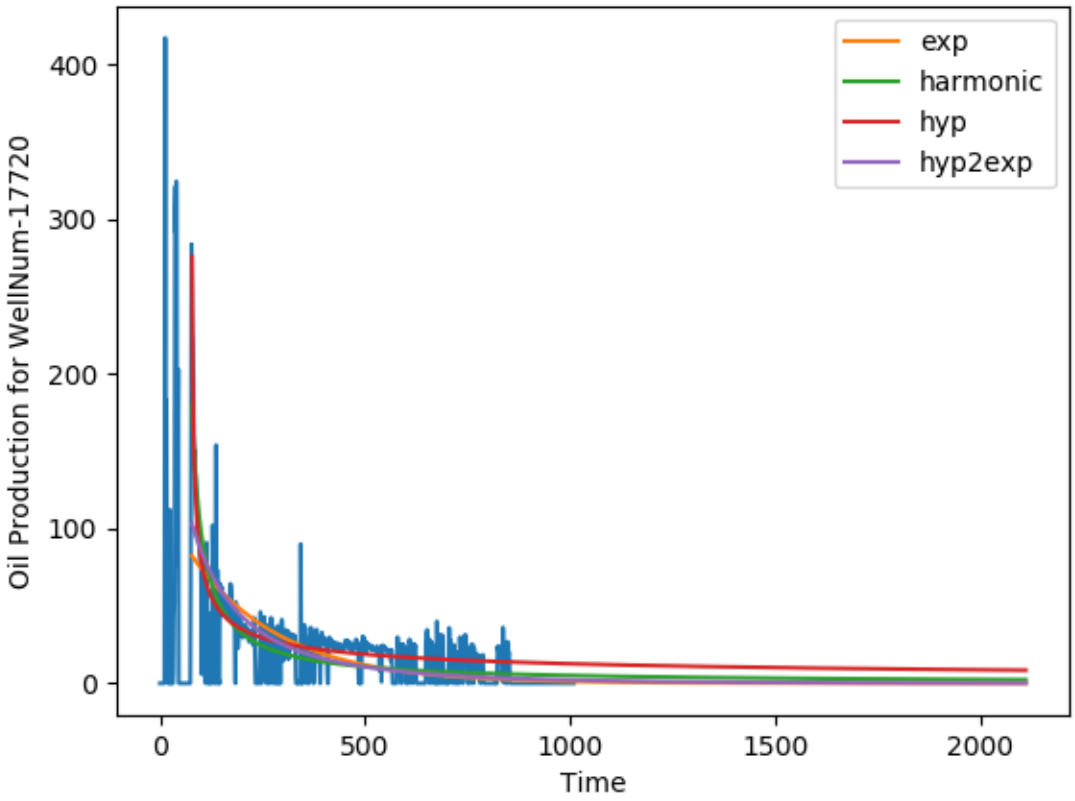
The best model is based on RMSE,AIC and BIC scores so it may be a little biased if there are outliers in data, which is the case most of the times. So, visualizing hyp2exp for oil and exp for gas is recommended. See test csv for these.

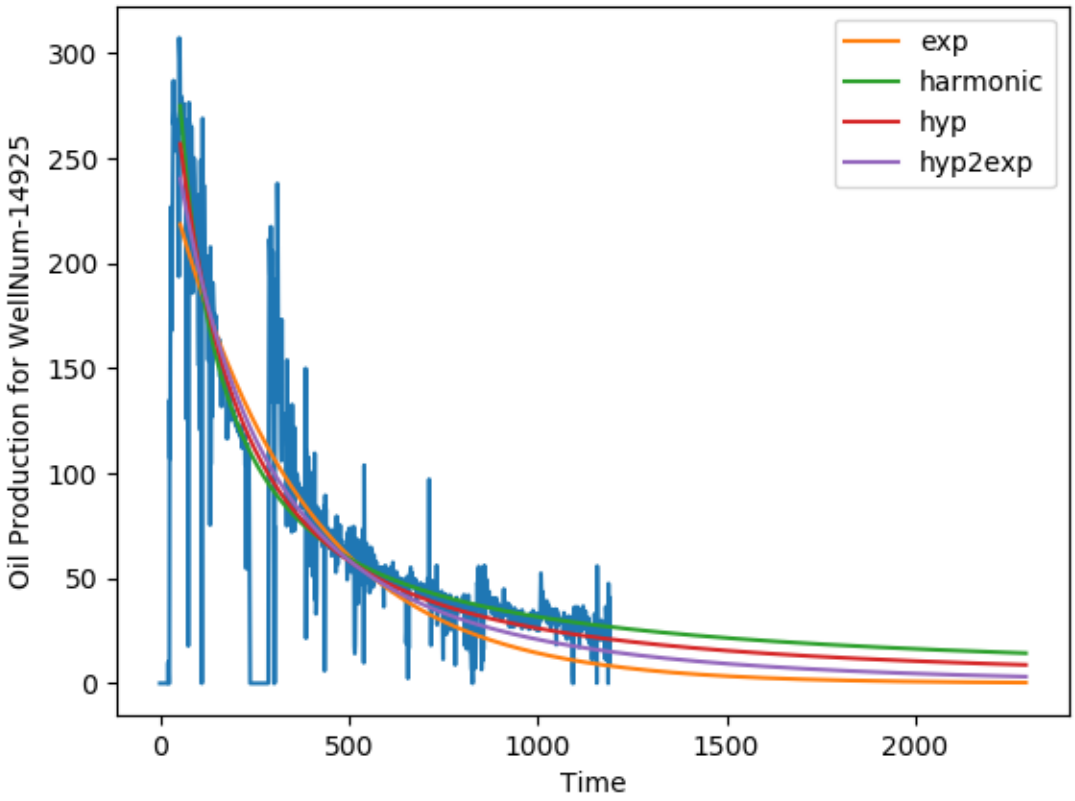
Also, use visualize.py for visualization purposes.

2.8. Forecast results for some of the wells




exp
harmonic
hyp
hyp2exp
Oil Production for WellNum-17647
0
100
200
300
400
500
0
500
1000
1500
2000
Time


exp
harmonic
hyp
hyp2exp
Oil Production for WellNum-17720
400
300
200
100
0
0
500
1000
1500
2000
Time



2.9. Unit testing

Unit testing is done in a more comprehensive manner covering all the functional units of the code. Please ignore the warnings as mentioned in the warnings section.

Pytest library is used. To validate the coverage pytest-cov plugin is used. A total coverage of 97% is achieved using this library.

2.10. Reproducing the results using python 3.6

Runtime: Python 3.6.1

See configuration.py to set the parameters for the code. Note that most of the parameters are already set according to the data. You can change the bounds for variables like b, beta, d and EUR.

Step by Step:

Copy the required train, test and api.list files both xlsx and csv into my submission folder.

- Install dependencies: pip install -r requirements.txt
- After the dependencies have been installed, run the main.py to fit and forecast.
- Run “python main.py” and wait until the wells are processed.

- Now, run “python evaluate.py” to run the evaluation pipeline and get evaluation results.
- Run “python fill_test_csv.py” to fill the test_ops_results_blank.csv with all the outputs.
- Now to visualize the data, use “python visualize.py”
- Run the unit tests with the following command.

Run pytest --cov=unit_tests/ --disable-pytest-warnings -v, python

- 97% code coverage.

2.11. Main Advantages of this approach

1. A comprehensive analysis of outlier algorithms is presented and the best one is chosen based on the data.

2. R code is converted to python and some enhancements were made to the approach by using machine learning.

3. All the fits and forecasts along with RMSE scores are good.

4. Multiprocessing is done to speed up the process.

5 .A visualization pipeline is set up.

6. The code is well written and commented following pep-8 format.

7. Other forecasting algorithms are also explored.

2.12. Feasibility of this approach

The method is quite simple to implement, and it takes only 80 seconds to train and forecast on 112 wells’ oil and gas decline curves. This implantation is highly tested using unit testing.

4. Other Forecasting algorithms to consider

LSTM: This is one of the best algorithms and works well for forecasting declines. https://www.onepetro.org/conference-paper/SPE-190104-MS

LSTM is an algorithm which highly depends on the size of the training dataset. As, the data size is good, our results on the testing data are good. Neural networks like Long Short-Term Memory (LSTM) recurrent neural networks can almost seamlessly model problems with multiple input variables. Use of ARIMA algorithm is also suggested. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

It is recommended to try out this curve fits in future challenges.

5. Appendix

5.1. References

1. Bilinear interpolation. https://en.wikipedia.org/wiki/Bilinear_interpolation
2. Piecewise linear function: https://en.wikipedia.org/wiki/Piecewise_linear_function
3. Milos Hauskrecht. “CS 2750 Machine Learning. Lecture 6. Linear Regression”. https://people.cs.pitt.edu/~milos/courses/cs2750-Spring03/lectures/class6.pdf
4. https://petrowiki.org/Production_forecasting_decline_curve_analysis
5. https://drive.google.com/file/d/1ggk8uwc5WvSzHUNuwlJEG2LgU2LWF80z/view?usp=sharing
6. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e
7. https://www.reddit.com/r/statistics/comments/5h2rbw/does_python_have_a_package_for_aicbic/
8. https://www.onepetro.org/conference-paper/SPE-190104-MS
9. https://www.onepetro.org/download/conference-paper/SPE-190104-MS?id=conference-paper%2FSPE-190104-MS
10. https://cs230.stanford.edu/projects_spring_2018/reports/8290385.pdf
11. https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

6. **Data Science Pipeline**

1. Retrain the model every month. If required, the retraining can be done frequently.

2. Use new data to retrain the model every week or month.

3. As new data comes in the model will become more accurate.

7. **Recommended Infrastructure to Run the Model**

Currently using the machine with following specifications:

1. RAM: 16GB

2. Cores: 4

3. Hard Disk: 500GB

Unit test results:

```
---------- coverage: platform win32, python 3.6.3-final-0 ----------
Name                                Stmts   Miss  Cover
-------------------------------------------------------------------------
unit_tests\decline_functions.py         9      0   100%
unit_tests\evaluate.py                 40      4    90%
```

unit_tests\identify_number_and_volume_of_peaks.py	18	1	94%
unit_tests\initial_estimate_of_time_to_peak.py	17	0	100%
unit_tests\outlier_analysis.py	13	0	100%
unit_tests\prepare_data.py	24	2	92%
unit_tests\signal_features.py	11	0	100%
unit_tests\test_decline_functions.py	15	0	100%
unit_tests\test_evaluate.py	29	0	100%
unit_tests\test_identify_number_and_volume_of_peaks.py	15	0	100%
unit_tests\test_initial_estimate_of_time_to_peak.py	15	0	100%
unit_tests\test_outlier_analysis.py	13	0	100%
unit_tests\test_prepare_data.py	9	0	100%
unit_tests\test_signal_features.py	10	0	100%
TOTAL	238	7	97%