Title

<https://www.kaggle.com/c/web-traffic-time-series-forecasting/>

Runtime: 13. Jul. – 16 Nov. || 1095 teams entered

**Data:**

* Train\_\* (1, 2) 145k X 551 / 804
* Key\_\* (1, 2) 8.7/8.99m / 2

**Data Description:**

* Training data consist of approx. 145k time series, each represent a number of daily views of a Wikipedia article. Keys denote the page of interest.
  + Each time series represent daily views from 01-07-2015 to 31-12-2016
  + Train\_1 is based on traffic from 01-01-2017 till 01-03-2017
  + Train\_2 is based on traffic until 01-09-2017.
  + Final Submission is based on predictions for period 13-09-2017 to 13-11-2017 for each article.

**Goal:**

* Predict Historical (Train\_1) and real (Train\_2) page views on Wikipedia articles

**Benchmark:**

* A

**Evaluation:**

* Symmetric Mean Absolute Percentage Error (SMAPE)
  + 
  + A\_t = Actual, F\_t = Forecast

**Summary:**

* Top 1-2 both use time-dependent medians as features
* Most solutions use time-dependent categorical, i.e. DOW, WOY, MOY, etc.
* Most solutions use context-dependent categorical, i.e. Page, Agent, Access, etc.
* Neural Networks performed best.
* All solutions are ensembles

# Contributions

**# 1/1 Place: Name Arthur Suilin, Language Py, Score (Pub/Pri) 35.48065**

* Repo: <https://github.com/Arturus/kaggle-web-traffic>
* Features:
  + Page views transformed by log1p
  + Agent, country, site one-hot encoded
  + Day of week
  + Year-to-year / quarter-to-quarter autocorrelation to capture seasonal strength
  + Page popularity (median of page views) to capture traffic scale
  + Lagged page views
    - Weighted average of important historical point, i.e. value last quarter or last year:
      * Attn\_365 = 0.25 \* day\_364 + 0.5 \* day\_365 + 0.25 \* day\_366
  + Preprocessing:
    - All normalized to zero mean unit variance (also one-hot)
    - Page views series are normalized independently
    - Stretch time-independent variables to fit length of TS
    - Model train on fixed-length samples of TS, but with random starting point
* Method:
  + Used RNN seq2seq model
  + Objective function: Smoothed Differentiable SMAPE
    - Percentage errors crash with zero/near-zero observations
  + Use COCOB optimizer in combination with gradient clipping.
    - Tries to estimate learning rate at each training step
  + Use walk-forward split 🡪 Hold-out set corresponding to test period 🡪 TimeSlice
    - Timeframe for validation is shifted forward by one prediction interval relative to timeframe for training.
  + Reduce prediction variance:
    - Find convergence region (here 1500-11500 epochs) save checkpoint every 100 epochs
    - Train 3 models with different seeds = 30 checkpoints
    - Use SDG Averaging on 30 checkpoints
  + Hyperparameter Tuning
    - Used SMAC3 for hyperparameter search

**# 2/2 Place: Name CPMP, Language Py, Score (Pub/Pri) 36.78499**

* Repo: <https://github.com/jfpuget/Kaggle/tree/master/WebTrafficPrediction>
* Features:
  + Project & access/agent are one-hot encoded
  + Medians of log1p transformed data
* Method:
  + Used Feed forward NN model
  + Objective Function: Custom Function
    - Error > 0 = 200, Error < 0 = -200, else = 0
  + Encoded output indices to w?\_d?. Third Thursday in same position independent of year.
    - Enabled extrapolation of weekly patterns
  + Network Configuration:
    - 200, 200, 100, 200 cells in each layer
    - Activation: RELU
    - Dropout = 0.5
    - Batch normalization for middle layer
    - Optimizer: ADAM
  + Used 5 fold CV with batch size 4096
  + Made 1 model for each fold, total 5 models and take median of their predictions.
  + Lastly, XGB is applied to NN residuals to extrapolate remaining info
    - Features:
      * WDAY, YDAY, NN Pred, Median Pred, Month, AllVisits, WeekEnd, Median/Mean + Ratio to Baseline, SiteLabel, FirstVal, AllVariance, AllMax

**# 3/3 Place: Name ThousandVoices, Language Py, Score (Pub/Pri) 36.85302**

* Repo: <https://gist.github.com/thousandvoices/7d01f366a388516359915a4b090e29d4>
* Features:
  + Medians over last 7, 28, 49, 365 days
  + Page visits same day last year
  + Median over same weekday
* Method:
  + Used CNN Model
  + Objective Function: SMAPE (from Wikipedia)
  + All data was normalized by division with median over entire training period
  + 4 convolutional layers, with configuration:
    - 32 filters, kernelsize=3 & 32 filters kernelsize=3, dilation\_rate=7
    - GRU layer with 256 hidden units

**# 4/4 Place: Name Chung Ming Lee, Language Py, Score (Pub/Pri) 36.86270**

* Features:
  + Log-transformed count data
  + DOW, WOY
  + Page project, access, agent
* Method:
  + Used Sliding Window Feed Forward NN with 120 days of look-back
  + Two Fully Connected Layers as Input tracks:
    - Direct Input of features
    - CNN preprocessing

**# 5/5 Place: Name Nathaniel Maddux, Language Py, Score (Pub/Pri) 37.13244**

* Log-transformation of response prior to fitting/prediction. MAE used as error measure
* Method: 4 Parts
  + (1) Simple, constant, median-based prediction
  + (2) Median for each day of the week
    - Based on last 35 days. Mean of medians was subtracted.
  + (3) Nonlinear polynomial autoregression
    - Past (X) = Medians of page views from 2, 4, 8, 16, 32, 64, 128 days in the past
    - Future (Y) are polynomial median regression coefficients of future page views.
      * In other words: Constant + trend 🡪 Y is predicted from X using 150 nearest neighbor
  + (4) Pages with strong yearly season
    - Strength is found by comparing value to last years value (after subtracting the mean from each list)
  + Model is combined with a linear crossover function:
    - F(x: x0, 01) =
      * If ( x0 <= x <= x1 ) == ( x - x0 ) / ( x1 - x0 )
      * If( x < x0 ) == 0
      * If ( x > x1 ) == 1
    - Activates median model when page views are low and yearly pattern model when pattern is strong.
    - Function is applied sperately to median page and early pattern models to get mixed portions. If there is any leftover ratio, it is assigned to the nonlinear polynomial regression model.
      * Mixing portions were normalized to sum 1.
    - Lastly, the weekly pattern was added to the prediction and exponentiated to invert the initial log transform.
  + Training data was generated by choosing a random date for each page that had 480 past days and 62 future days (correspond to length of test set)
  + 10-fold cross validation was used.
  + Tuning done with 2 passes of line searching, and prior manual tuning
  + Final model is a 10x ensemble, where the main source of variation is the random stop day chosen for each training set.

**# 6/6 Place: Name sjv, Language Py, Score (Pub/Pri) 37.42000**

* Repo: <https://github.com/sjvasquez/web-traffic-forecasting>
* Used a stack of dilated causal convolutions.
* Model was trained using next step prediction, so errors accumulate as the model generates long sequences in the absence of conditioning information. Loss was then reset every 64 steps.