Title

<https://www.kaggle.com/c/walmart-recruiting-sales-in-stormy-weather>

Runtime: 2. Apr. – 19 May. ||485 teams entered

**Data:**

* Key.csv – Mapping between stores and weather stations
* Test.csv -
* Train.csv
* Weather.csv

**Data Description:**

* Historical sales for 111 products (milk, bread, umbrellas, etc.) sold in 45 different locations
* 45 locations are covered by 20 weather stations
* Weather events defined by any day in which more than an inch of rain or two inches of snow is observed.
* Sales = 0 does not necessarily mean that there was no demand, but could mean that item was in stock without demand, or out of stock, or discontinued and not available.
* Train/Test fields:
  + Date, store\_nbr, station\_nbr, item\_nbr, units, id (store\_item\_date)
* Features included:
  + Max temp, min temp, avg temp, depart, dewpoint, wetbulb, heat, cool, sunrise, sunset, codesum, snowfall, preciptotal, stnpressure, sealevel, wind speed, wind direction, avg wind speed

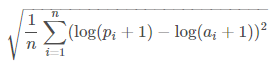
**Goal:**

* Predict units sold for a window of +- 3 days surrounding each storm

**Benchmark:**

* ??

**Evaluation:**

* Root Mean Squared Logarithmic Error (RMSLE)
  + 
  + N = nrow
  + P = predicted units
  + A = actual units
  + Log(x) = natural logarithm

**Summary:**

* Date features are very important
* Fluctuating sales around events, e.g. black Friday
* As.numeric(Date) improves performance for ML
* Weather features did not help significantly
* Ensemble models always outperform single models

# Contributions

**# 4/1 Place: threecourse, Language: R/Py, Score (Pub/Pri): 0.09379/0.09339**

* Preprocessing
  + Exclude item/stores with all zeros
  + Apply curve fitting in R ppr (lop1p\_units, from 20112-01-01)
  + Exclusions:
    - 2013-12-25 🡪 Mostly zeros
    - Moving average (21 elements, centered) is zero
* Method:
  + Train Linear model with vowpal wabbit
    - Y = log1p\_units – ppr\_fitted
  + Prediction:
    - Exp(Ppr\_fitted + linear model predicted)
* Features Used:
  + A: Weekday, is\_weekend. Is\_holiday\_and\_weekday, is\_holiday\_and\_weekend
  + B: Item\_nbr
  + C: Store\_nbr
  + D: Date
  + E: Year, month, day
  + F: Is\_blackfriday-3days, -2days, -1day || Is\_blackfriday+1day, +2days, +3days
  + G: Weather
    - Preciptotal > 0.2
    - Depart > 8
    - Depart < -8
  + Interactions
    - A\*B, A\*C, B\*E, C\*E, B\*F, C\*F
* Conclusions:
  + Weekdays most important
  + Month periodicity
  + Black Friday indicator
  + Weather features are not effective 🡪 People shop no matter the weather.

**# 11/3 Place: Timothy Scharf, Language: R, Score (Pub/Pri): 0.09490/0.09417**

* Features:
  + Weather (Numeric) 🡪 All Features (Except CodeSum)
    - Actual/day, 7day mean prior, 7day mean post
  + Date (Numeric)
    - As.numeric(Date) 🡪 Vector from Day 1 to Day N
    - Day1:7, month, Day1:365, Day1:31, Day1:1035
* Method:
  + 1000 XGBOOST models (linear average == .098 score)
    - Random parameters for each model
    - Y = log1p(y)
    - Early stopping
    - Save OOB indexes
  + Boost from prediction with new features:
    - New Feats 🡪 OOF, week day (cat), month (cat)
    - Smaller learning rate (around 100)
  + Conclusions:
    - Using As.numeric(Date) helped immensely
      * Also adding date features as numeric allows for fast tuning
    - First model poor at estimating weekday fluctuations & month fluctuations
      * Hence included categorical features for these in model 2

**# 6/5 Place: Daniel Korzekwa, Language?, Score (Pub/Pri): 0.09415/0.09428**

* Method: Bayesian Regression 🡪 Gaussian Process
  + Single Gaussian Process for each store/item pair + sharing hyperparameters
    - likelihood in a log scale
    - Features:
      * Model 1:
        + Epoch time, day of week, first half of month
      * Model 2
        + Epoch time, day of week, day of month, quarter of year
      * Storm tommorw && (no stort for today and for last three days)
  + Final Model: Ensemble of Model 1 and Model 2
* Conclusions:
  + Tomorrow’s storm might be more important than todays.

**# 8/6 Place: Little Boat, Language?, Score (Pub/Pri): 0.09415/0.09428**

* Features:
  + Set 1:
    - Date variables (week, month, weekday, ets.)
    - Weather features (all chars converted to zero)
      * Omitting codeSum, sunrise, sunset
  + Set2:
    - Date variables
    - Historical sales (ignoring first year)
      * Sale same day last year
      * Sale from week last year
      * Average sale in month last year, etc.
* Method:
  + Complex models (RF, SVM) on Set 1 features
  + Simple models (linear regression, ridge regression) on set 2
    - Except RF, models were applied per store per item
  + Per store per item, time series model (e.g. ARIMA) to predict sales
  + Predict 0 for store with historical zero sales.
    - Group by item to predict sales from other stores (where its not raining) to predict missing ones
  + Ensemble of all models (about 30 models in total)
* Conclusions:
  + Ensemble increase score immensely
  + Ridge regression on Complex + Simple Models provide score of 0.95XX

**# 19/11 Place: vtKMH, Language: Py, Score (Pub/Pri): 0.09621/0.09490**

* Preprocessing:
  + Combine train and test
  + Zero out event periods
  + Drop store/item combos with zeros or out-of-season
  + Fill in event periods with combinations of front-/back-fill
  + Drop non-event periods
  + Split back into train and test
* Features:
  + Mean, Median, Max, STD, Percent zero, mean of non-zero, median of non-zero, etc.
    - On rolling horizons: 30, 14, 7, 5, 3
  + Store-item-weekday average / 7 day mean of units for non-event periods
  + All weather features
    - Did not add much
* Method:
  + Ensemble of 6 boosted trees, 4 extremely randomized trees, 1 KNN, 1 linear model.
    - Sklearn
  + Used mostly same features in all models, but changed it a little by subtracting rolling means/medians from units or other roling features
* Postprocessing
  + Zeroed prediction for Christmas day
  + Manually calculated Thanksgiving adjustments
* Conclusions:
  + RF contributed little to overall model
  + ExtraTrees performed very well (EXTremely RAndomized Tress)
  + Poor results with NNs