Review at least 10 interviews, bounce around, do not go sequentially.

1) What model(s) did they use?
2) Insights;
3) Feature creation, selection;
4) Other observations.

1, Xavier Conort

- Uses pretty standard algorithms. Random Forests. Gradient Boosting Machines.
- Most of the effort is spent on feature engineering. Careful to discard features that lead to over-fit.
- Spends a lot of time exploring and visualizing the data to understand what brings value.
- Works in a team of 40+ data scientists in Singapore

2, Ben Hammer

- For a time series, took N trailing components as features.
- Used random forests
- Changing the error metric and optimizing for it directly
- Use data to decide how many trailing components to use

3, Tim Salimans

- Used Bayesian analysis
- Used a custom formula for posterior distribution. Change the error metric
- Look at intermediate data (viz, distribution). Change subsequent steps based on that.
- Very noisy, messy data. Bayesian seemed to help with that.
- Gradient-based optimization and random restarts (avoiding local minima)

4, James Petterson

- Used general boosting regression models
- Didn't look at the data much
- Used R
- Used a little bootstrapping to increase size/variety of data (24-hour moving window)
- Surprised that he did so well without more feature extraction or intermediate analysis

5, Matthew Carle

- Had 35 entry submissions

6, Marcin Pionnier

- Surprised by the power of boosting weak learnings.
- Used an ensemble of different models.
- Some feature extraction from the initial data.
- Used boosted decision stumps
- Used Weka and Java

7, Vlado Boza

- Used a fairly clean data set
- Was able to identify what would happen in different Kaggle competitions - ensemble fights, random forests, too much effort given bad data, hyper-competitive, etc
- Used a 2-layer neural network
- First attempt used k-NN

- Used C++, a little Python
- Not much feature extraction

9, Illdefons Magrans

- Tried SVM, LR, GBM, RF. Finally stayed with random forests
- This was a difficult challenge
- Used R and a laptop

10, Daniel McNamara

- Used only things that proven strong using cross-validation
- Random forests.
- Classification tree boosting
- Regression tree boosting
- Neural network
- Ensembled them together
- A simple data set with few features means the competition is fierce. Ensembling and teaming up are required.
- Neural netowrks worked really well here
- Used SQL, SAS, R, even Excel

11, Bo Yang

- Best result was an ensemble of random forests, gradient boosting machines, and 2 forms of logistic regression
- Did a lot of feature extraction using Google Maps API and JavaScript. However, it didn't help much
- Most useful variables were fairly well known
- Used SQL, R, C++, C#, JS, Google, Excel

12, Dyankonov Alexander

- Tried to split into 2 different predictors without success
- Used kernel density estimator and probability
- Used Matlab

Contest: Neil Schneider on Dunnhumby Shopper Challenge

Models: GBM (Gradient Boosting Machine)
Insights: Shoppers are habitual, Predict Spend on a per weekday per customer basis
Feature Selection: None
Other Observations: Time Series models for inventory control performed poorly. Regression fit the model poorly because evaluation metric minimizes distances from quartiles.

Contest: William Cukierski on DunnHumby Social Challenge

Models: logistic regression,
Insights: Very local dataset. Predicting far out in future is useless. Used mode where usually mean would be used.
Feature Selection: PCA, SVD, #visits, mode time since visit, each day has a different feature matrix,
Other Observations: Features based on polling friends didn't help. Big list of things that didn't work out.

Contest: Dell Zhang on long live wikipedia

Models: Gradient Boosted trees

Insights: future behavior can be determined by recent behavioural dynamics
Feature Selection:
Other Observations: dynamic features go a long way when we choose right temporal periods and machine learning methods

Contest: Keith Herring on long live wikipedia

Models: Ensemble of Randomly constructed decision trees (Breiman and Cutler)
Insights: most informative features captured both timing and volume of edits
Feature Selection: 206 features derived from editor attributes like timing, volume, automation, commenting etc
Other Observations: wrote a web scraper to get Wikipedia editing history,

Contest: Xavier Conort in 'Give me some credit'

Models: Mix of 15 models, best model was ensemble without SVM
Insights: likelihood of being late is very important
Feature Selection: created a variable to estimate likelihood of being late by more than 90 days
Other Observations: data science is helped by diversity, ensemble of weak models gave better performance

Contest: Joe Malicki in 'Give me some credit'

Models: Random Forests, Gradient Boosted trees, Logistic regression, and SVMs
Insights: # times past due is a good feature, noting that DTI for those with missing income data was critical
Feature Selection: expanded feature set using apriori knowledge of credit scoring and personal finance
Other Observations: stratified sampling to produce training set, random forests perform bad when transforms of features are needed, fraud is likely if income is divisible by 1000 , two level model (second model trained on probabilities of individual models – choose best per class model, cross validation is important and trust worthy

Contest: Tuzzeg in 'Impermium'

Models: logistic regression for basic classifiers and random forest for ensembles
Insights: sentence level features, many insulting posts were one sentence, bigger posts had just one insulting sentence
Feature Selection: stem based features, parser results and POS tag features,
Other Observations: simple stem based features (sub sequences and n grams) worked better in final model

Contest: Eu Lin in 'Merck Challenge'

Models: SVM and Random forests
Insights: success factor is selecting the significant variables,
Feature Selection: ran GBM on a small data sample and used that to select features
Other Observations: dataset is highly variable, used a gradient increase as a feature selector,

Contest: Rouli Nir in 'Recommendation Engine Challenge'

Models: Random forest, gradient boosting classifier, and some naïve bayes
Insights: iterative approach to guessing user location and event location
Feature Selection: top feature was time between when user was presented with event and event date, % of users invited but didn't attend, total attendees, and estimated distance between user and event

Other Observations: over fitting happens, don't believe random forest hype, always use random seeds,

Contest: Guocong Song on ' Salary Prediction'

Models: ridge regression, SGD, Random Forests,, naïve bayes, SVM, logistic regression (SGD + Logistic- final model)
Insights: models that can explore local properties would outperform linear regression
Feature Selection: typical text stuff like normalization, stop words, n grams,TF-IDF,
Other Observations:

Contest: Vlado Boza, on 'Salary Prediction'

Models: Nearest neighbor model, Neural network with 2 small hidden layers
Insights: similar ads with salary differing by 2000
Feature Selection: simple binary text features from title and description, categorical features for company, location and source
Other Observations: good bugless coding is important

What models did they use?

Vlad Boza (2nd place, job salary prediction) - "old school" neural network
Guocong Song, 3rd Prize, Job Salary Prediction Competition – blend of SGD regression and logistic regression
1st place Observing Dark Worlds – Bayesian model
3rd in Merck – SVM's, gradient increases, neural networks ensemble
Troll-hunter competition 2nd – ensemble of logistic regression and random forest
Wordpress Challenge winner – ensemble of GLM + random forest
Air quality prediction – random forest
Don't get kicked competition – neural nets & GBM. Neural networks used only for the weighting coefficient in the blending model.
Claim Prediction – Clustering and neural networks
Trading challenge (3rd place) – K nearest neighbors; lasso regression

Insights

Vlad Boza (2nd place, job salary prediction) – neural network handled similarity in clusters of the data better than nearest neighbor model he started with
1st place Observing Dark Worlds – interesting that a physics problem can be a 'traditional' data science/ statistics problem
3rd in Merck – tried a ton of different models and ensembles to find the right combination
Wordpress Challenge winner – used processing power rather than 'elegance' to solve the problem
Air quality prediction – no prior knowledge of the data
Don't get kicked competition – relationship between prices is much more informative than the prices themselves
Trading challenge – ensemble of reasonably good models performs better than a finely tuned individual model

Feature creation

Vlad Boza (2nd place, job salary prediction) – extracted binary text features from title and description and used categorical features for location, company, source
Guocong Song, 3rd Prize, Job Salary Prediction Competition – used extensive text feature extraction techniques such as text normalization, stop words, n-grams
1st place Observing Dark Worlds – fixed certain variances at a maximum value
3rd in Merck – used GBM's to select the right features

Troll-hunter competition 2nd – used simple stem based features (subsequences and n-grams); worked better than complex features
Wordpress Challenge winner – 'wrote ugly python scripts to create data features'; one of the key features was one designed to represent node centrality, a measure of how important a node is within a social graph
Air quality prediction – used lagging N components from the full time series as features
Don't get kicked competition – changed textual values to numerical format; created a secondary synthetic variable comparing prices/ costs.
Trading challenge – feature set picked by hand using feedback from the training set; most were basic transformations (such as spreads between values rather than absolutes)

Other observations

Guocong Song, 3rd Prize, Job Salary Prediction Competition – needed a model that could 'explore local properties' (i.e., not a linear regression)

1st place Observing Dark Worlds – incredible that a simple model can capture a lot of natural complexity
3rd in Merck – pre-processing and modeling was a big challenge/ benefits of multiple approaches
Troll-hunter competition 2nd – insulting internet posts tend to be one sentence in length
Wordpress Challenge winner – feature selection critical to winning the competition (unique representation of node)
Air quality prediction – no domain knowledge necessary
Don't get kicked competition – feature selection with regard to relationships between variables was critical to success
Claim Prediction – examining data clustered by household (claims contest) proved to be very important
Trading challenge – one member joined after taking a free Stanford Machine Learning class; skewed error distribution; found that the market was unpredictable in the opening hours (8am) and this time frame contributed a disproportionate amount of error; trained a different model for the market open (naïve benchmark model); found that the bid/ ask spread followed separate paths depending on who initiated the trade – so, performing regression separately for each category led to dramatic improvement in prediction accuracy. One team member noted that the knowledge of what was going on under the hood made a difference in improving his model – and that the best model of their ensemble was written without any 3rd party libraries and was able to fit the quirks of the trading data. Also noted the power of a team to improve insights.

Competition: Merck 1st place

Model(s): Neural networks and Gaussian process regression
Insights: There were lots of similarities between the 15 tasks involved and using all of the inputs provided were valuable, even when they were intended for a different task. Ridge regression did not provide a meaningful improvement over equally-weighted model averaging.
Feature creation/selection: Deliberately avoided most feature creation tasks, some log-transform of input features
Other Observations:

Competition: Practice Fusion Diabetes Classification – 1st place

Model(s): GBM, Random Forests and stacking in a generalized additive model
Insights: Many comorbidities and symptoms occur with diabetes. Diabetes is much more prevalent in men than women, possibly because of other comorbidities.
Feature creation/selection: Translated each medication to its active principles and route of administration. Grouped principles together based on domain knowledge. Grouped diagnoses in base CCS and domain knowledge.

Other observations:

Competition: Predicting a Biological Response

Model(s): Random Forests, GBM, KNN, Neural Networks, Elastic Net, GLM, GAM and SVM.
Insights: Obtaining an accurate ranking of the relative importance of the variables and eliminating some was critical. Clusters of points became visible when looking at data through principal components.
Feature creation/selection: Random Forests
Other observations: Multiple people with different perspectives are even more important than the ensemble of algorithms.

Competition: Arabic Writer Identification Challenge – 1st place

Model(s): Linear Discriminant Analysis
Insights: When working with a small training and test set, generalization is difficult
Feature creation/selection: All features used as provided
Other observations:

Competition: Deloitte/FIDE Chess Rating Challenge – 1st place

Model(s): Modified Glicko system (based on a Bayesian model)
Insights: Iteratively updated ratings schemes can be very competitive relative to systems that would not be applicable in practice
Feature creation/selection: None explicitly noted
Other observations:

Competition: Give Me Some Credit – 1st place

Model(s): Random forest of classification trees, random forest of regression tress, a classification tree boosting algorithm, a regression tree boosting algorithm and a neural network
Insights: Feature creation was critical for predictive capability
Feature creation/selection: Number of late days and difference between income and expense.
Other observations: Neural networks performed surprisingly well

Competition: Dunnhumby Shopper Challenge – 2nd place

Model(s): GBM
Insights: Shoppers are habitual about when they do most of their shopping
Feature creation/selection: Breaking data out by weekday was critical
Other observations: GBM is powerful, but the coefficients for independent variables can be meaningless

Competition: Mapping Dark Matter – 1st place

Model(s): Fit minimization engine and Neural Network
Insights: Pixel level residuals were a powerful tool. Neural networks got worse when centroid position was passed.
Feature creation/selection: KSB measurements of ellipticity, ellipticity corrections, maximum-likelihood fit of images, convolving galaxy and point spread function models
Other observations: There was a very tight cluster of scores at the top of the leaderboard with wildly different methods.

Competition: Predict Grant Applications – 2nd place

Method(s): Ensemble Selection method using AdaBoost, LogitBoost, RealAdaBoost, DecisionTable, RotationForest, BayesNet, NaieveBayes.
Insights: Decision rules for grant applications changed in 2007/2008.

Feature creation/selection: Many simple transformations, such as dates to year, month and day components.
Other observations:

Competition: Job Salary Prediction Competition – 2nd place

Method(s): Neural network with 2 small hidden layers
Insights: Ads were very similar
Feature creation/selection: Binary text features from title and description, categorical features for location, company and source
Other observations: Versioning is valuable, even when working on something like a kaggle c

INFORMS Data Mining Contest

1) Logistic regression, neural networks and SVM
2) Insight - preprocessing of data was very significant
3) Feature selection - forward stepwise selection, reverse stepwise selection
4) Other observations - created additional features during preprocessing, and the use of 5-folds cross validation.

Elo Comp

1) Home-grown variant of "Ensemble Recursive Binary Partitioning"
2) Insights - created various combinations of predictors
3) Variable selection - synthesized a variety of predictor variables by observing model performance both on the leaderboard set and holdout set
4) Other observations - of the 120 submission, the 93rd produced the best score

Kaggle-in-Class competition at Stanford

1) Linear model, random forest and the winning one was k-NN (k-Nearest Neighbor, with k=1)
2) Insights - they would not have thought of using k=NN model without carefully studying and analysing the data first
3) Feature selection - ignored continuous variables (~80% of feature) and utilizes only the nominal variables as explanatory variables.

IJCNN Social Network Challenge

1) Neural networks, random forests, boosting and SVMs.
2) Insights - ended up using random forests because it yielded the best AUC mark and required no data preprocessing.
3) Feature selection - programmed 20 features (Jaccard, Adar, Katz, various neighbor-related metrics, path lengths, unseen bigrams, etc) along with a framework whereby extraction of local "neighborhood graphs" to make the O(N2) computations feasible.
4) Other observations - combine techniques and features from other competitors and ran 100's of random forest algorithm to determine the feature with the most signal

Dunnhumby's Shopper Challenge

1) Supervised regression methods, gradient based approaches
2) Insights - customer's recent shopping behavior was more predictive (has more signal) than his past behavior
3) Feature selection - Weighted kernel density estimate

4) Other observations - carefully analyzing and optimizing according to the evaluation metric can dramatically affect which models performs better.

Mapping Dark Matter Challenge

1) Artificial neural network
2) Insights - pixel-level residuals are a powerful tool for finding the best-fit model for galaxy and star images.
3) Feature selection - non-obvious subset of the likelihood output
4) Other observations - despite the wide variety of methods applied at or near the top of the leaderboard, the scores are tightly clustered.

Photo Quality Prediction competition

1) Mix of random forest, GBM and logistic regression
2) Insights - use of many raw & derived variables from external location-based data. Most useful were simple and well known albums, weighted with global average based on how many albums the word appeared in. No signal was obtained out of word pairs.
3) Feature selection - Calculated additional features
4) Other observations - Use of SQL, R, C++, C# and Javascript for Google web services API.

Don't Get Kicked

1) Regression models, bagging algorithms, decision stump and boosting
2) Insights - No data transformation or data generation, most important insight was to choose algorithms that aggregate many weak learners efficiently
3) Feature selection - No external data sources needed, the creation and addition of some interpreted variables values to the initial dataset was however necessary.
4) Other observations - boosting weak learners ends up to be very powerful in terms of performance. Linked Weka library with Java code for data preprocessing.

Arabic Writer Identification Challenge

1) LDA, which was popular and successful in face recognition ten years ago.
2) Insights - training and test data are of small size, therefore had to be careful about generalization of the model ability
3) Feature - used provided features with linear discriminant analysis
4) Other observations - must be careful about overfitting in real word problems as oppose to academic research.

Biological Response

1) Random forest, GBM, KNN, Neural network, Elastic Nets, GLM, GAM, and SVM.
2) Insights - plotting the data points using 2nd and 3rd principal components did not help, however obtaining accurate ranking of the relative importance of the each variables was very important, also recognizing danger of overfitting.
3) Feature selection - eliminating variables (sequentially?)
4) Other observations - tools used were R and Matlab.

1st place interview Boehringer Ingelheim Biologicall Response (7/5/2012, Jeremy Achin)

Models/methodologies:

Random forests for feature ranking and selection
K-NN, neural network, SVM, elastic net

Tools:

R and Matlab

Insights:

Feature ranking very important to eliminate variables
Watch out for overfitting
Diverse team important (multiple viewpoints)

Meet the Winner of the Algo Trading Challenge: An Interview with Ildefons Magrans (January 26 2012, Ildefons Magrans)

Models/methodologies:

Tried SVM, logistic regression, GBM, random forests – ended up with random forests

Tools:

R

Insights:

Complex training set, different samples could fit different models
Not surprised by difficulty level since it involved high-speed trading (lots of smart people)

The Perfect Storm: Meet the Winners of 'Give Me Some Credit' (Jan 3 2012, Daniel McNamera)

Models/methodologies:

Random forest of classification trees, random forest of regression trees, classification tree boosting algorithm, regression tree boosting algorithm, neural network

Tools:

R, SAS, SQL, Viscovery, Excel

Insights:

Neural networks performed well for this competition, better than bagging and boosting.
Three individuals with different skill sets (algorithms, credit scoring, data mining) came together to win the competition.

Creatures of Habit: Neil Schneider on placing second in the dunnhumby Shopper Challenge (October 20, 2011 Neil Schneider)

Models/methodologies:

Generalized boosting regression, histograms, partition models

Tools:

SAS, JMP (data visualization for SAS), R

Insights:

Shoppers are habitual, usually have similar spending habits by day of week, so splitting out projections by weekday was helpful. Regression models did not work, possibly due to the test statistic used for evaluation; regressions needed to be optimized for the highest density areas.

DeepZot on Dark Matter: How we won the Mapping Dark Matter challenge (September 13, 2011 Daniel Margala)

Models/methodologies:

Neural net, maximum likelihood estimation, fit minimization

Tools:

C++, ROOT data analysis framework (http://root.cern.ch)

Insights:

Tight cluster of scores at the top of the leaderboard despite wide variety of methods applied – all these methods appear to be making use of the maximum amount of information in the images in the presence of pixilation and noise.

Like Popping Bubble-Wrap: Keith T. Herring on Placing Second in Wikipedia's Participation Challenge (October 6, 2011 Keith T. Herring)

Models/methodologies:

Random forest, decision tree, parameter randomization, ensembling

Tools:

Python, MySQL, C++, Matlab

Insights:

Feature extraction was key, final predictor operated on 206 features derived from timing, volume, name-space contributions, etc. Most informative features captured both edit timing and volume of an editor, with exponential weighting. External data such as community political dynamics would be helpful.

Mind Over Market: The Algo Trading Challenge 4th Place Finishers (January 26, 2012 Will Cukierski)

Models/methodologies:

K-NN, linear regression, lasso regression, SVM, random forest, K-means

Tools:

R, C++, Matlab

Insights:

Market open accounted for large proportion of RMSE so separate models were used for different times of day; recognizing special events (liquidity shock trade) forced revising model around those events. Also, splitting out bid and ask prices showed different regression models. Fewer, better tuned models better than throwing the kitchen sink at the problem.

Picture Perfect: Bo Yang on winning the Photo Quality Prediction competition (November 23, 2011 Bo Yang)

Models/methodologies:

Random forest, GBM (geometric Brownian motion?), logistic regression

Tools:

R, SQL, C++, C#, Javascript, Excel, Google web services

Insights:

Don't try to find a single key to the data ("one ring to rule them all"), stick with simple, well known variables. No signal obtained from word pairs. Don't try to add lots of extraneous data, spend time refining algorithms and included variables.

Smile! Alexander D'yakonov on placing third in the Photo Quality Prediction competition (November 27, 2011 Alexander D'yakonov)

Models/methodologies:

Random forest, weighted K-NN.

Tools:

Matlab, R

Insights:

No special insights, just some extra features such as image aspect ratio and area, along with a merged word list. Surprised that there were no good features in the word lists – all engineered features were worse than expected.

On Diffusion Kernels, Histograms, and Arabic Writer Identification (April 29, 2012 Yanir Seroussi)

Models/methodologies:

SVM (linear, polynomial, RBF and sigmoid kernels)

Tools:

Python with libSVM and SVMLight

Insights:

Lack of correlation between cross-validation results on the training data and the accuracy on the validation set, probably due to the conditions of the test. Using different a different kernel based on character histograms yielded the most significant performance boost.

Competition and team: Merck Molecular Activity, 3rd, '.'

What models did they use

- *EJ:* For models, I tried everything from GBMs to PLS but it came down to just SVM and Random Forest.
- *AL:* I used SVMs, gradient increases and neural networks to build several models which was subsequently put together to create the final submission.

Insights

- The key success factor was selecting the significant variables, and for this I used a gradient increase as a feature selector.

- I created a glmnet model that used sparse matrix representations of each data set. Unfortunately, my approach did not crystalise to a strong solution.
- Alex discovered that a GBM run on a sample of the data could be used to select features and greatly speed up the full model.
- I was surprised by Alex's approach wherein he ran a GBM on a sample of the data which he then used to select features. Did not expect that to work well but it did so that was an insight for me.
- Yea me too! But also the temporal effect of the dataset, which was prevalent in the activity of the molecules.

Feature creation, selection

- *EJ:* I used SVD to reduce features which was used as training data.
- *AL:* The key success factor was selecting the significant variables, and for this I used a gradient increase as a feature selector.

Other observations

- *AL:* I was surprised by the coincidence of errors in different parts of the test suite (public error, private error).
- Tools: *Together:* R. *AL:* And open office too. *EJ:* And excel too, for graphs.
- *EJ:* … try everything and don't give up. Every tiny effort you put in will bring you closer to the top.
- *ZM:* RAM is cheap and you should have a lot on your prototyping machine! I personally couldn't afford to keep an m2.4xlarge EC2 instance running for a month or 2...
- *AL:* The benefits of multiple approaches.

Competition and team: Observing Dark Worlds, 1st,

a. What models did they use

- TS: Bayesian analysis provided the winning recipe for solving this problem:
    - Construct a prior distribution for the halo positions $p(x)$, i.e. formulate our expectations about the halo positions before looking at the data.
    - Construct a probabilistic model for the data (observed ellipticities of the galaxies) given the positions of the dark matter halos: $p(e|x)$.
    - Use Bayes' rule to get the posterior distribution of the halo positions: $p(x|e) = p(e|x)p(x)/p(e)$, i.e. use to the data to guess where the dark matter halos might be.
    - Minimize the expected loss with respect to the posterior distribution over the predictions for the halo positions: $\hat{x} = \arg\min_{\text{prediction}} \mathbb{E}_{p(x|e)} L(\text{prediction}, x)$, i.e. tune our predictions to be as good as possible for the given error metric.

b. Insights

- Step 2 is more complicated. Fortunately the competition organizers provided us with a set of training skies for which the positions of the dark matter halos was known, as well as a summary of the physics behind it all. After reading through the tutorials and forum posts it became clear that the following model should be reasonable:
  $p(e_i|x) = N(\sum_{j=\text{all halos}} d_{i,j} m_j f(r_{i,j}), \sigma^2)$

c. Feature creation, selection

- None mentioned

d. Other observations

- the outcome of this competition was more noisy than is usual: final prediction accuracy was judged on a set of only 90 cases, with an evaluation metric that is very sensitive to small (angular) perturbations of the predictions. The public leaderboard standings were even more random, being based on only 30 cases. In fact, the 1.05 public score of my winning submission was only about average on the public leaderboard. All of this means I was very lucky indeed to win this competition.

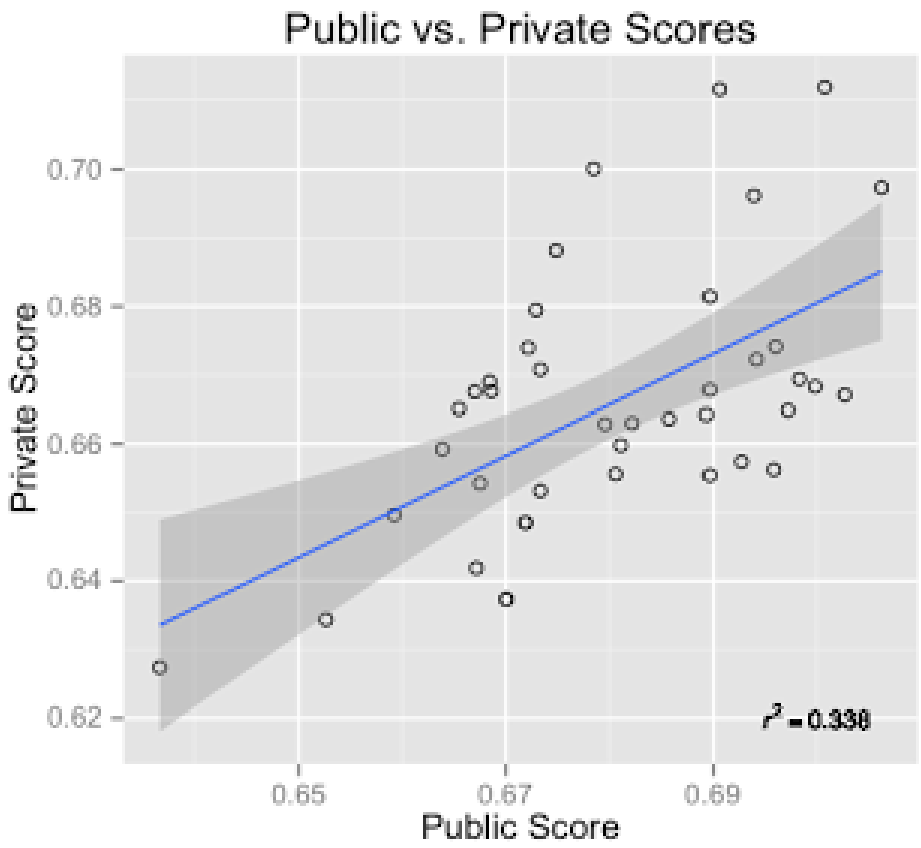
Competition and team: Recommendation Challenge, 3rd, Rouli Nir

a. What models did they use

- I've treated the challenge as a classification problem, and ranked the events by their predicted probability to be classified as interesting.
- My model is basically an average between a Random Forest and a Gradient Bosting Classifier, with a sprinkle of Naive Bayes on the events' descriptions.

b. Insights

- Over Fitting is a Bitch: As you can see in the graph below, the public score was a very bad predictor for the private (and final) score.



Luckily, I've trusted more my five-fold validation averaged score than the scores I got on the public leader board. But boy, was that frustrating. A few times during the competition I've submitted my predictions after fixing some bug or improving the reliability of some feature just to get a lower public score. Even worst, in the last couple of days before the board freeze, I was unable to improve my public score and was watching new-comers getting far better scores than mine. I was exiled from the warm comfort of the fourth place to the 11th place. It wasn't easy to ignore the public score and not to optimize against it, but luckily I did so.

- Lesson Two: Don't Believe Random Forest's Hype
  - It was my first time successfully employing a random forest classifier as the main predictive mode (usually, logistic regression works better for me), and I believed all the hype about random forests being better at avoiding over fitting. However, the telltale this isn't the case here was observing that adding features to the model sometimes decreased my validation score.
  - I've combated that behavior by limiting the trees in the random forest to a certain maximum depth and a minimum number of samples at each leaf, and averaging with the GB classifier (which I haven't tweaked).

    - Knowing what I know now, I would have also dropped some features that I've added to the model just because they seemed relevant and I believed Overkill Analytics' approach of adding as many features as possible to a random forest, and let it sort them out.
- Lesson Three: There's Always Some Data Leakage
    - Midway through the competition I've discovered a feature with a very strong predictive power. Turns out that in many cases you could guess whether a user in the train set is interested in a specific event just by looking at the timestamp when he observed that event. If a given user observed several events at timestamp X, and another one at timestamp Y>X (even if those are just a few seconds apart), that other "later" event was probably marked as interesting.
    - Obviously this was some sort of a bug in the train set, and once I've proved that it was exploitable by reaching the third spot, I've (foolishly, since I didn't get any "finders fee") alerted Kaggle about it. [Kaggle note: Much appreciated!!] Fortunately, I got back to third place without such dirty tricks.
    - However, in the way the train and test sets were assembled, there was some leakage that could (and should) be exploited. First, for each user we had a list of about six events, from which exactly one was marked as interesting. That is, we are given a lot of information in comparison of the classical classification problem. I've exploited that by having a feature that compared the delta of each event displayed to a user against the delta of the earliest event. This proved to have more predictive power than the number of friends a user had in the event.
    - Moreover, we can assume that startup behind the competition already propose only relevant events to its users. For example, users from Indonesia were rarely presented with events happening in the US. They probably do this by examining IP address which they chose not to share in the train set.
    - This led to my first attempt and still most successful at creating a "distance between user and event" feature; I simply calculated the median of the locations of the events presented to the user, and for each event calculated its distance from that median, as though the median was a substitute to the user's location. This worked better than deriving the user coordinates by looking at events occurring at the same city as the one found in the user's profile, and averaging between them.
- Lesson Four: Always (Always!) Use Random Seeds
    - Some of my features are derived from graphs, such that a pagerank score for events according to their attendance graph (where events share an edge with a weight that depends on the Jaccard similarity between the users that attended them), or, like many other did, events and users clusters. Sadly, because my hardware isn't powerful enough, I had to prune the graphs and I did so by deleting edges at random.
    - Big mistake. Now I cannot recreate my final submission, and after all the work I put into it, I may not see the prize money. I was smart enough to set a random seed for my random forest, but too lazy or stupid to that when creating the graph. Gah!

c. Feature creation, selection

- The top feature for me (as judged by the random forest), was the time between when the user was presented with the event and the event's date, or delta for short. Next come the ratio of invited users that decided not to attend an event (surprisingly, the higher the ratio, the more interesting is the event), the total number of attendees, and the estimated distance between the user and the event (more on this later).

d. Other observations

- RN: I strive to learn from each competition I participate in, and this one is no different. However, my take-aways from this challenge don't involve a new algorithm or feature selection insights. Rather, they are lessons about handling data on Kaggle challenges and in real life.
- Tools: I used pandas, which I really wanted to try out (great, but I'm missing some of R's data exploration methods), scikit-learn and iPython notebook (fantastic!).

Competition and team: Job Salary Prediction, 2nd, Vlado Boza

a. What models did they use

- My whole model was just an old-school neural network with two small hidden layers trained by back propagation. Before that I used nearest neighbor model which was quite successful (got error around 4200).

b. Insights

- During one point I found out that there are too many similar ads and that their salary differs on average by 2000. I used this in my nearest neighbor model. But neural network could handle this even better without any hacks.
- Ad similarity was the only thing.

c. Feature creation, selection

- I extracted simple binary text features from title and description and also used categorical features for location, company, and source.

d. Other observations

- Tools: I have coded all of my algorithms in C++ (I did small preprocessing in Python). I tried to use scikit-learn but it didn't lead to any big success.
- I have to improve my coding practices. I've made many stupid bugs just because of this. And I also should start to use some versioning system better than “do backup sometimes”.

Competition and team: Merck Molecular Activity, 2nd, DataRobot

a. What models did they use

- Methodologies explored for various roles include Random Forests, PCA, GBM, KNN, Neural Nets, Elastic Net, GAM, and SVM.

b. Insights

- For most problems, GBM and SVM had similar predictive power and produced similar predictions. However, for problem 5, SVM's predictions deviated significantly from GBM's predictions and scored badly on the public leaderboard. This confirmed the importance of having at least 2 very different models in one's toolkit. We also improved the accuracy slightly by capping the predictions of a few problems.
- The most surprising thing was that almost all attempts to use subject matter knowledge or insights drawn from data visualization led to drastically worse results. We actually arranged a 2 hour whiteboard lecture from a very talented biochemist and came up with some ideas based on what we learned, but none of them worked out.
- Also, the visualizations that were shared as part of the visualization part of the competition were incredible (thanks to all who contributed!). We drew much insight from them which led us to try some new approaches that we were absolutely sure would work. However, most of these approaches failed to improve results, and many of them drastically decreased our public leaderboard scores. The visualizations did make us take a second look at capping which helped a bit.
- We were also surprised to see how well our internal CV scores correlated with the public (and private) leaderboard scores. This was unexpected because of all the evidence suggesting the test sets were very different from the training sets for some problems. Only for problem 4 did the public leaderboard give us faulty feedback, but since problem 4 was so small, we knew to include a submission that ignored the public leaderboard feedback and included our best CV model for problem 4.

c. Feature creation, selection

- See (a).

d. Other observations

- Tools: We used R, Python, and a lot of computing power. We really didn't start working on the problem until around 2 weeks before the deadline, so we had to cram lots of cpu cycles into a short amount of time. We used our 9 Ubuntu servers, Amazon, plus Xaviers magic macbook which he somehow gets to perform like it is a 32 core machine with 256GB of RAM. I (Jeremy) was sure the thing would combust at any moment, so I made sure all the smoke alarms in the house had fresh batteries.
- We also used airplanes--Xavier came to the US and worked with us in person for 4 days which allowed us to get a good head start on the problem and make a solid two week plan (which we stuck to for the most part).

Competition and team: CPROD, 1st, ISSSID

a. What models did they use

- Supervision learning: We employed "Conditional Random Field" Model. We choose this algorithm because it converges faster and is easy to implement. We used tool MALLET for this purpose.

b. Insights

- The specific characters of products naming (Example: iPhone – mixture of both uppercase and lowercase) and human sematic behavior analysis (Example: my iPhone or <action> by <company name>) are the most important insight that helped us to improve the precision overall. We finally took voting approach (to find which category the product belongs to) based on our experimental results.
- When we merged the Conditional Random Field Model to other two models (Standard and Rule Template) we have, the performance we achieved significantly increased. We got approximately 3% improvement in F1 score.
- Notably combinations of CRF models achieved the highest score in the private leaderboard, but not in the public leaderboard.

c. Feature creation, selection

- Preprocessing: (1) JSON format to plain text format (2) cleaning the data by deleting all useless characters and symbols. (3) Change all uppercase for few products to lowercase

d. Other observations

- Languages: C++, Python, Perl
- Tool: Mallet

Competition and team: Diabetes Classification; Practice 1st – 3rd; blind ape (Jose), mtb (Matt), and An apple a day (Sashi)

a. What models did they use

- Jose: the methods used were the well known gbm and randomforest and later stacking in a generalized additive model.
- Matt: All of the models I selected for my final submissions were boosted trees. I used anywhere from 5 to 13 different models and blended/combined their predictions to create my submissions.
- Shashi: I used random forests, gradient boosting and neural networks to build several models which were finally stacked together to generate a final solution.

b. Insights

- Jose: The great numbers of comorbidities and symptoms associated with diabetes.

- Matt: The ICD9 data is rich. There is information in the hierarchy of the codes (i.e. what level a specific code belongs to), information regarding the health of the individuals family (i.e. any of the 'family history of' codes) and information regarding the individuals behavior (i.e. any of the 'history of non-compliance' codes). And of course the conditions that are associated with each of the codes. For the first month or so of the competition I focused almost solely on feature generation. And a majority of these features were derived from the ICD9 codes.
- Shashi: One big insight was that the formats for some key fields were different in train and test datasets. This was causing a much larger error on test data than that on training data. I could not think of any explanation for this for quite a while. Fixing the formatting discrepancy resolved this issue and made the train and test errors consistent.
- Jose: I'm surprised by gender overall impact. In the data, diabetes was much more prevalent in male (15%) than female (11%) but when fitting the model, the gender influence fell to 0.17%. Probably other comorbidities associated to gender would explain this reduction.
- Matt: I was most surprised by the area's that I did not find interesting features. I did not find the lab data very useful and I thought the drug information would be more useful than it was. That surprised me. I would be interested in seeing what features the other competitors found in this area.
- Shashi: I created several new features by taking ratios of different pairs of features. I was surprised by the extent to which these features improved my model. I created these features towards the very end of the competition. It helped me jump up a couple of places on the leaderboard.

c. Feature creation, selection

- Jose: Main work was data cleaning and feature creation. Grouping diagnostics and actives principles was crucial. As an advance of my solution, I based it in a hard preprocessing and feature creation work:
    - Translating each medication to its active principles, route of administration.
    - Grouping principles active by chemical families / clinical indication. In some cases, as statins, adjusting dose equivalences. Choosing for each group between number of prescriptions, dose or binary flag for feature creation.
    - Grouping the diagnoses in base CCS and my personal experience.
- Matt: I spent a fair amount of time generating features from the ICD9, NDC and lab data. I used wikipedia heavily to learn more about diabetes and create features from the diagnoses and treatments that are related to diabetes.
- Shashi: The preprocessing was limited to missing value imputation for a few fields in the data tables.

d. Other observations

- Jose: R
- Matt:
    - .Net / C# for writing the logic that generates the features

    - SQL Server for storing the data as well as basic analysis (just sql queries from management studio)
    - Python and scikit-learn for training, testing and evaluating the models
    - R for graphical analysis (ggplot2)
- Shashi: I used only R to do all the data processing and the modeling. Excel was used just a little bit to do some quick plots on the data.
- Jose: I learned a lot about active principles and their interactions with diabetes.
- Matt: Trust your cross validation scores and use the public leaderboard as a measurement of competitiveness. When I selected my final models for submission, I picked the models with the lowest cross-validation scores, not the ones with the lowest public leaderboard scores. This worked well for me in this competition, using this formula, I ended up picking my five best models.
- Shashi: The most important learning was that feature engineering is of utmost importance. Perhaps even more than any fine-tuning of modeling algorithms. Allocating a lot of time to just review the data and create useful features can result in significant performance improvement.

Competition and team: Boehringer Biological Response, 1st, Winter is Coming (Tom, Jeremy, Sergey)

a. What models did they use

- Methodologies explored for various roles included Random Forests, GBM, KNN, Neural Nets, Elastic Net, GLM, GAM, and SVM. Simple transformations & splines were used for some models.

b. Insights

- I would say that the most important insight into the data was obtaining an accurate ranking of the relative importance of the variables. Eliminating variables reduced model training time (allowing us to try more things) and improved performance considerably.
- Another important insight was recognizing the danger of overfitting inherent in this problem. This led us to design a testing framework that helped to ensure we didn't overfit the public leaderboard.
- By plotting the data points using the 2nd and 3rd principal components, you can see 4 very separated clusters of points. We thought this was going to be a very important finding, but it didn't turn out to help us.

c. Feature creation, selection

- We used Random Forests for feature ranking & selection.

d. Other observations

- R & Matlab. Rstudio Server is awesome if you are using multiple computers--I love having a browser open with many tabs each interfacing with a different RStudio Server.
- First and foremost, having a team with diversity and relentless tenacity is extremely important. I'll quote Shea Parkes because I couldn't possibly phrase it any better: "It's quite obvious how much an ensemble of viewpoints can contribute above and beyond an ensemble of algorithms."
- The ability to collaborate effectively is critical. We exchanged 349 emails over the 45 days that we worked on the competition, and we were in sync enough to be able to share, test, and improve on each other's work. At one point, Sergey took over 17,000 files we generated using R and combined them with his own results in Matlab allowing us to reliably test out many different blends.
- Also, if you are going to run long jobs (20+ hours), make sure to put a sign up in your house to remind people not to plug an iron into the same circuit your computers are on. I went into shock when all of a sudden the monitors went black and the computers went silent. It was like someone yanked the Matrix data probe out of the back of my head.

Competition and team: CHALEARN Kinect Gesture Challenge, 1st, Alfonso Nieto-Castanon

a. What models did they use

- I did not implement any learning strategy but used instead a combination of ad hoc features from the depth videos (somewhat inspired by neural processes in the visual system) with a Bayesian network model for recognition.

b. Insights

- Thinking of gestures as a form of communication, and realizing that the subjects in those videos were already doing what they thought would work best in order for us to interpret and recognize those gestures correctly. I imagined that a system that would mimic the specificities of the human visual system would be most likely to pick up those helpful cues from the video sequences correctly.

c. Feature creation, selection

- None mentioned.

d. Other observations

- I used Matlab (just the matlab base set, no specific toolboxes other than the nice set of functions provided by the contest organizers to browse the data and create a sample submission)
- I enjoy developing problem-specific algorithms rather than using a combination of off-the-shelf procedures. This contest gave me the chance to do just that while working in one of

those (few) areas where humans still outperform machines (and I am curious to see if we can further bridge that gap!)

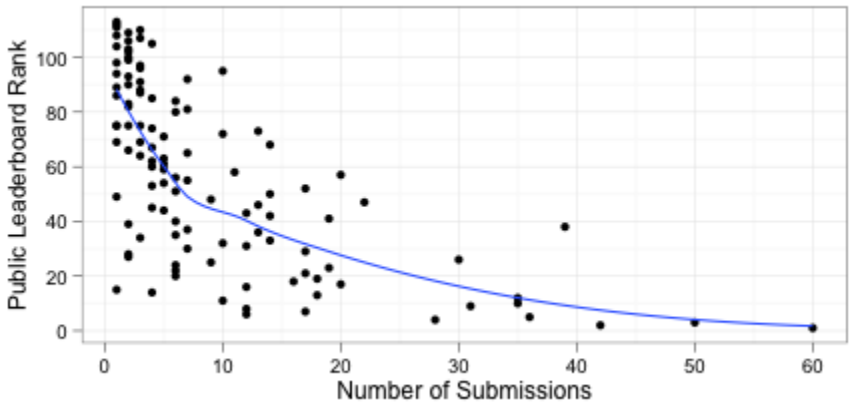
Competition and team: The Dangers of Overfitting or How to Drop 50 spots in 1 minute; 52nd; Greg Park

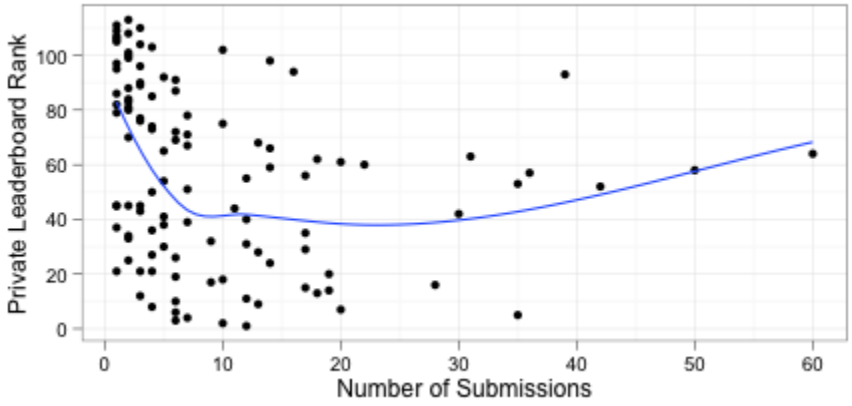
a. What models did they use

- Not mentioned.

b. Insights

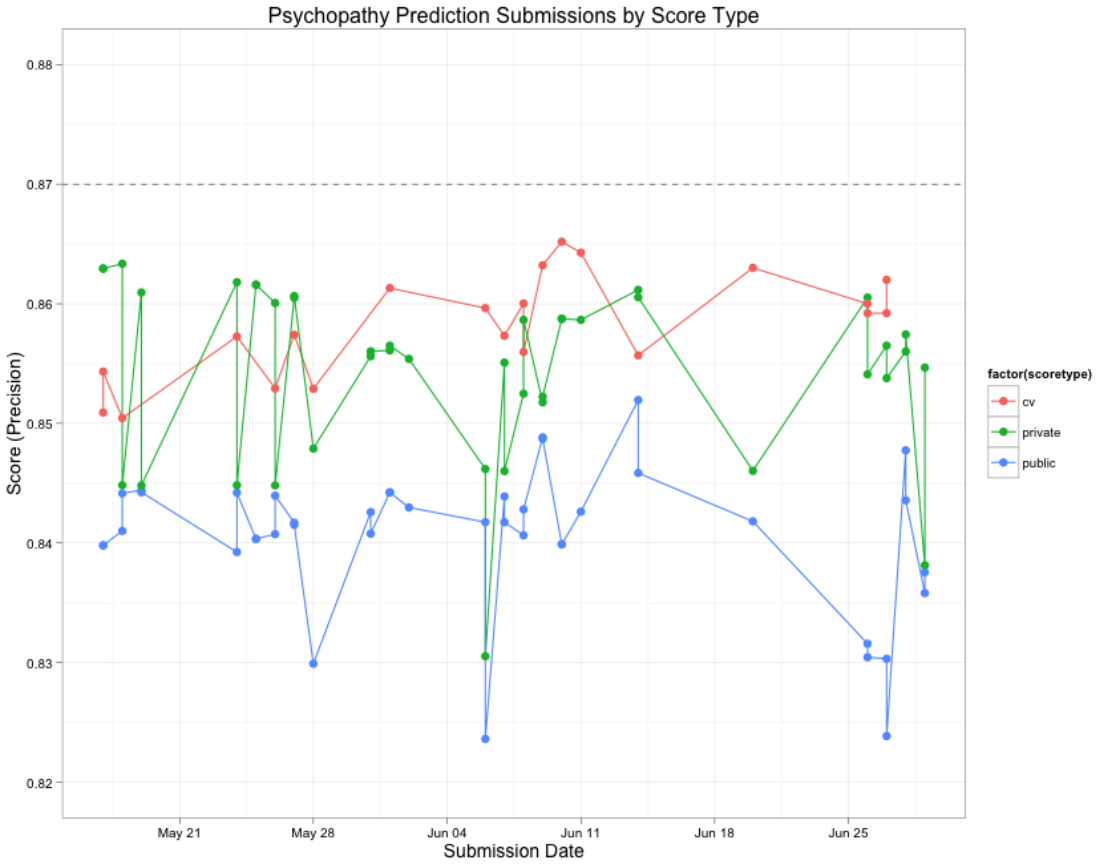
- Over the course of the contest, I made 42 submissions, making it my biggest Kaggle effort yet. Each submission is instantly scored and competitors are ranked on a public leaderboard according to their best submission. However, the public leaderboard score isn't actually the "true" score – it is only an estimate based on a small portion of the submission. When the competition ends, five submissions from each competitor are compared to the full set of test data (all 1,172 Twitter accounts), and the highest scoring submission from each user is used to calculate the final score. By the end of the contest, I had slowly worked my way up to 2nd place on the public leaderboard,
- I held this spot for the last week and felt confident that I would maintain a decent spot on the private or "true" leaderboard. Soon after the competition closed, the private leaderboard was revealed.
- Where'd I go? I dropped from 2nd place on the public leaderboard to 52nd on the private leaderboard. I wasn't the only one who took a big fall: the top five users on the public leaderboard ended up in 64th, 52nd, 58th, 16th, and 57th on the private leaderboard, respectively. I even placed below the random forest benchmark, a publicly solution available from the start of the competition.
- After getting over the initial shock of dropping 50 places, I began sifting through the ashes to figure out what went so wrong. I think those with more experience already know, but one clue is in the screenshot of the pack leaders on the public leaderboard. Notice that the top five users, including myself, have a lot of submissions. For context, the median number of submissions in this contest was six. Contrast this with the (real) leaders on the private leaderboard – most have less than 12 submissions. Below, I've plotted the number of entries from each user against their final standing on the public and private leaderboards and added a trend line to each plot.





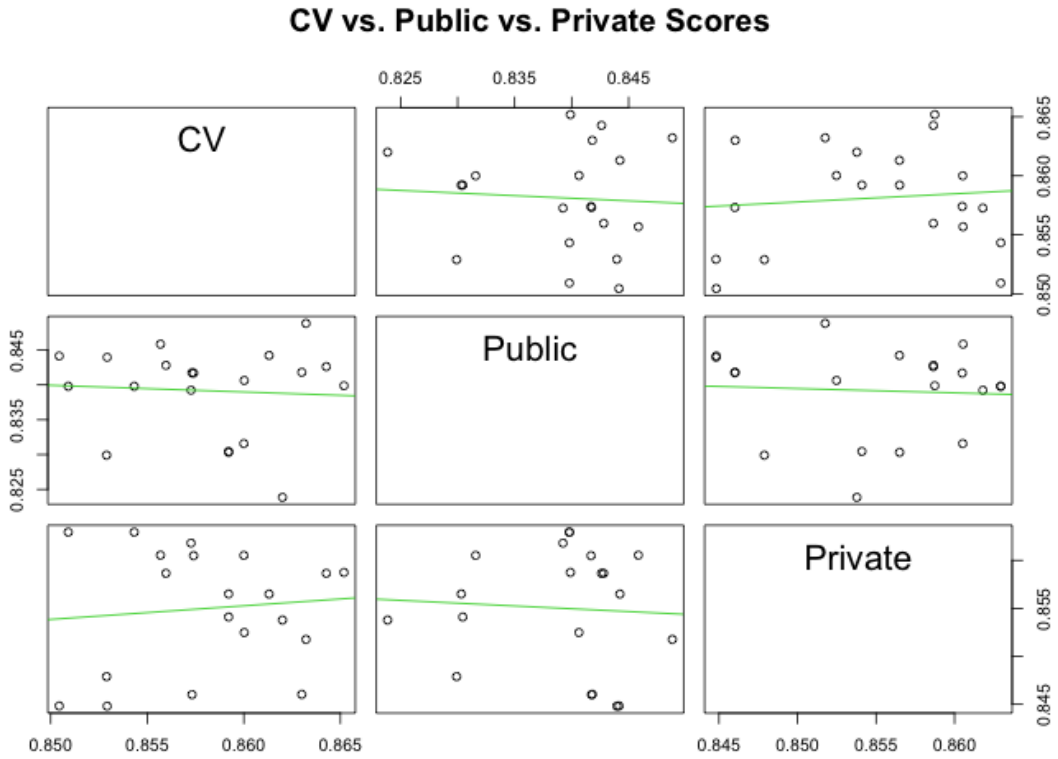
- On the public leaderboard, more submissions are consistently related to a better standing. It could be that the public leaderboard actually reflects the amount of brute force from a competitor rather than predictive accuracy. If you throw enough mud at a wall, eventually some of it will start to stick. The problem is that submissions that score well using this approach probably will not generalize to the full set of test data when the competition closes. It's possible to overfit the portion of test data used to calculate the public leaderboard, and it looks like that's exactly what I did.
- Compare that trend in the public leaderboard to the U-shaped curve in the plot of the private leaderboard and number of submissions. After about 25 submissions or so, private leaderboard standings get worse with the number of submissions.
- Overfitting the public leaderboard is not unheard of, and I knew that it was a possibility all along. So why did I continue to hammer away at competition with so many submissions, knowing that I could be slowly overfitting to the leaderboard?
- Many competitors with estimate the quality of their submissions prior to uploading them using cross-validation. Because the public leaderboard is only based on a small portion of the test data, it is only a rough estimate of the true quality of a submission, and cross-validation gives a sort of second opinion of a submission's quality. For most of my submissions, I used 10-fold cross-validation to estimate the average precision. So throughout the contest, I could observe both the public leaderboard score and my own

estimated score from cross-validation. After the contest closes, Kaggle reveals the private or "true" score of each submission. Below, I've plotted the public, private, and cv-estimated score of each submission by date.



- Scores of my 42 submissions to the Psychopathy Prediction contest over time. Each submission has three corresponding scores: a public score, a private score, and a score estimated using 10-fold cross validation (cv). The public score of a submission is calculated instantly upon submission by comparing a subset of the predictions to a corresponding subset of the test data. The private score is the "true" score of a submission (based on the entire set of test data) and is not observable until the contest is finished. Every submission has a public and private score, but a few submissions are missing an CV-estimated score. The dotted line is the private score of the winning submission.
- There are a few things worth pointing out here:
    - My cross-validation (CV) estimated scores (the orange line) gradually improve over time. So, as far as I know, my submissions are actually getting better as I go.
    - The private or "true" scores actually get worse over time. In fact, my first two submissions to the contest turned out to be my best (and I did not chose them for my final five)
    - The public scores are reach a peak and then slowly get worse at the end of the contest.

    - It is very difficult to see a relationship between any of these trends.
- Based on my experience with past contests, I knew that the public leaderboard could not be trusted fully, and this is why I used cross-validation. I assumed that there was a stronger relationship between the cross-validation estimates and the private leaderboard than between the public and private leaderboard. Below, I've created scatterplots to show the relationships between each pair of score types.



- Scatterplots of three types of scores for each submission: 10-fold CV estimated score, public leaderboard score, and private or "true" score.
- The scatterplots tell a different story. It turned out that my cross-validation estimates were not related to the private scores at all (notice the horizontal linear trends in those scatterplots), and the public leaderboard wasn't any better. I already guessed that the public leaderboard would be a poor estimate of the true score, but why didn't cross-validation do any better?
- I suspect this is because as the competition went on, I began to use much more feature selection and preprocessing. However, I made the classic mistake in my cross-validation method by not including this in the cross-validation folds (for more on this mistake, see this short description or section 7.10.2 in The Elements of Statistical Learning). This lead to increasingly optimistic cross-validation estimates. I should have known better, but under such uncertainty, I fooled myself into accepting the most self-serving description of my current state. Even worse, I knew not to trust the public leaderboard, but when I started to edge towards to the top, I began to trust it again!

- In the end, my slow climb up the leaderboard was due mostly to luck. I chose my five final submissions based on cross-validation estimates, which turned out to be a poor predictor of true score. Ultimately, I did not include my best submissions in the final five, which would have brought me up to 33rd place – not all that much better than 52nd. All said, this was my most educational Kaggle contest yet. Here are some things I'll take into the next contest:
    - It is easier to overfit the public leaderboard than previously thought. Be more selective with submissions.
    - On a related note, perform cross-validation the right way: include all training (feature selection, preprocessing, etc.) in each fold.
    - Try to ignore the public leaderboard, even when it is telling you nice things about yourself.

c. Feature creation, selection

- None mentioned.

d. Other observations

- Some sample code
- One of my best submissions (average precision = .86294) was actually one of my own benchmarks that took very little thought. By stacking this with two other models (random forests and elastic net), I was able to get it up to .86334. Since the single model is pretty simple, I've included the code. After imputing missing values in the training and test set with medians, I used the gbm package in R to fit a boosting model using every column in the data as predictor. The hyperparameters were not tuned at all, just some reasonable starting values. I used the internal cross-validation feature of gbm to choose the number of trees. The full code from start to finish is below:

```
library(gbm)

# impute.NA is a little function that fills in NAs with either means or medians
impute.NA <- function(x, fill="mean"){
  if (fill=="mean")
  {
    x.complete <- ifelse(is.na(x), mean(x, na.rm=TRUE), x)
  }
  if (fill=="median")
  {
    x.complete <- ifelse(is.na(x), median(x, na.rm=TRUE), x)
  }
  return(x.complete)
}
data <- read.table("Psychopath_Trainingset_v1.csv", header=T, sep=",")
testdata <- read.table("Psychopath_Testset_v1.csv", header=T, sep=",")

# Median impute all missing values
```

```
# Missing values are in columns 3-339
fulldata <- apply(data[,3:339], 2, FUN=impute.NA, fill="median")
data[,3:339] <- fulldata

fulltestdata <- apply(testdata[,3:339], 2, FUN=impute.NA, fill="median")
testdata[,3:339] <- fulltestdata

# Fit a generalized boosting model

# Create a formula that specifies that psychopathy is to be predicted using
# all other variables (columns 3-339) in the dataframe

gbm.psych.form <- as.formula(paste("psychopathy ~",
                                   paste(names(data)[c(3:339)], collapse=" + ")))

# Fit the model by supplying gbm with the formula from above.
# Including the train.fraction and cv.folds argument will perform
# cross-validation

gbm.psych.bm.1 <- gbm(gbm.psych.form, n.trees=5000, data=data,
                      distribution="gaussian", interaction.depth=6,
                      train.fraction=.8, cv.folds=5)

# gbm.perf will return the optimal number of trees to use based on
# cross-validation. Although I grew 5,000 trees, cross-validation suggests that
# the optimal number of trees is about 4,332.

best.cv.iter <- gbm.perf(gbm.psych.bm.1, method="cv") # 4332

# Use the trained model to predict psychopathy from the test data.

gbm.psych.1.preds <- predict(gbm.psych.bm.1, newdata=testdata, best.cv.iter)

# Package it in a dataframe and write it to a .csv file for uploading.

gbm.psych.1.bm.preds <- data.frame(cbind(myID=testdata$myID,
                                         psychopathy=gbm.psych.1.preds))

write.table(gbm.psych.1.bm.preds, "gbmbm1.csv", sep=",", row.names=FALSE)
```

"Troll Detection with Scikit-Learn"

Available at: http://blog.kaggle.com/2012/09/26/impermium-andreas-blog/

1) Used both an n-gram word and n-gram character model and finally an ensemble but the n-gram character model was best
2) A simpler model actually outperformed more complex model
3) Features were created for the number and ratio of words from a "bad" list published by Google and then modified by author for words specifically prominent in the training set
4) There were problems with the noise in the data, the author felt that all the top fifteen models were all within the standard deviation of his model.

"ASAP Interview with Marin O'Leary"

Available at: http://blog.kaggle.com/2012/05/13/asap-interview-with-martin-oleary/

1) Final model was an ensemble of various simpler models
2) In this case, the ensembles outperformed any individual simple model
3) Features were largely based on syntactic features in the data. Interestingly, grammar and spelling were tried but were surprisingly unimportant
4) In this completion the choice of error metrics was very important because some algorithms are specifically designed for certain error measures

"How I did it: Lee Baker on winning Tourism Forecasting Part One"

Available at: http://blog.kaggle.com/2010/09/27/how-i-did-it-lee-baker-on-winning-tourism-forecasting-part-one/

1) Weighted ensemble model taking advantage of three different models: 1) simple naive predictor, 2) polynomial line fit, 3) least square regressor
2) Very interesting use of domain knowledge (found that tourism was growing on average from a paper and included that knowledge)
3) Features selection was not a huge component since this was a fairly basic time series
4) One interesting problem in this contest was that the writer found that the last four years of data was not effective at predicting the leaderboard. His speculation is that this might be due to the global economy slump during the last few years

"How we did it: Jeremy Howard on winning the tourism forecasting competition"

Available at: http://blog.kaggle.com/2010/11/24/how-we-did-it-jeremy-howard-on-winning-the-tourism-forecasting-competitoin/

1) A weighted regression model
2) The data was pre-processed to remove outliers beyond three standard deviations
3) Features selection was not a huge component since this was a fairly basic time series
4) The author adapted the model so that the last observation was always well forecast. This was intended solely to increase the score on the MASE error metric

"Phil Brierley on winning tourism forecasting part two"

Available at: http://blog.kaggle.com/2011/03/22/phil-brierley-on-winning-tourism-forecasting-part-two/

1) An ensemble of two other ensembles. The first was a weighted average of ARIMA, ETS, Damped HoltWinters and a Seasonal Naïve model. The second ensemble was composed of ARIMA, ETS and Damped HoltWinters models trained on four different test data windows, plus a Naïve Seasonal model trained with all the test data. The forecast for the second ensemble was then just the mean of all thirteen of these models
2) The focus of this team was not to improve the overall accuracy of the model but rather to try to avoid the places where the benchmark ETS model was extremely wrong, so the ensemble models were intended to bring those outlier forecasts closer in towards the average
3) Features selection was not a huge component since this was a fairly basic time series
4) One interesting observation was that weekly tourism data might have been more accurate since tourism is particularly sensitive to events that are mostly on weekends. Therefore, comparing months from previous years may be difficult if one had five weekends while another only had four.

"How I did it: Yannis Sismanis on Winning the first Elo Chess Ratings Competition"

Available at: http://blog.kaggle.com/2011/02/08/how-i-did-it-yannis-sismanis-on-winning-the-elo-chess-ratings-competition/

1) A fairly simple logistical regression with regularizations to weigh recent games more and to account for the total number of games a player has played
2) Regularization was considered extremely important as it helped to avoid over fitting
3) A very interesting feature considered was an average ranking of opponents, based on the assumption that players or relatively similar strength often play each other

"Could World Chess Ratings be decided by the 'Stephenson System'?"

Available at: http://blog.kaggle.com/2012/03/20/could-world-chess-ratings-be-decided-by-the-stephenson-system/

1) An optimized form of the Glicko algorithm which is itself based on a Bayesian Model
2) In this case, novel models were not needed, just optimizations of an existing well performing model
3) Since the data set contained only results of games, there was not much mention of feature selection
4) One of the benefits specifically mentioned for this model was that it provides an advantage over the existing model without adding much complexity. Performance is of course a factor that would play into almost any real world data science applications.

"Mind Over Market: The Algo Trading Challenge 4th Place Finishers"

Available at: http://blog.kaggle.com/2012/01/26/mind-over-market-the-algo-trading-challenge-4th-place-finishers/

1) This team used a blended model of three different participant's models to achieve a 4th place finish. The models used were extremely diverse and included kNN, linear regression, SVMs and random forests
2) Domain insights were very important. For example, it was very hard to predict the first minute of a market day (in one model, the first minute of the day accounted for 12% of the total error).
3) Besides for time of day mentioned above, one of the important features was whether a transaction was initiated by the buyer or seller, since this had a huge impact on the final performance
4) An interesting insight of one of the participants was that they should have spent more time tuning some of the individual models and also ensuring that the right models were inside the ensemble model. Because of a time crunch, the team ended up putting all their models into a single ensemble instead of trying to prune the contents.

"Meet the Winner of the Algo Trading Challenge: An Interview with IIdefons Magrans"

Available at: http://blog.kaggle.com/2012/01/26/meet-the-winner-of-the-algo-trading-challenge-an-interview-with-ildefons-magrans/

1) The model used was a Random Forest
2) Since the dataset was from stock market trading, it was extremely volatile so different models were actually better fits on different subsets of the data
3) The interview doesn't mention any feature selection
4) In this dataset, even small differences in model performance can make a small difference

"Like Popping Bubble-Wrap: Keith T. Herring on Placing Second in Wikipedia's Participation Challenge"

Available at: http://blog.kaggle.com/2011/10/06/like-popping-bubble-wrap/

1) A Random forest with trees chosen for their prediction power. 3000 random models were created but only the best were chosen for the final model
2) Diversity was essential to building a well performing random forest
3) There were a number of interesting features found in this data including the edit volume of a user as well as the time between edits
4) This competition used raw data, so much like many real world data science projects, the processing and cleaning of data was very important

Mapping Matter with Matlab: Sergey Yurgenson on finishing second in Mapping Dark Matter http://blog.kaggle.com/2011/09/02/mapping-matter-with-matlab/

Model used: neural network
Insights: he started using the modified quadruple moments formula and fitting procedure, but being unsatisfied with the result kept coming back to neural networks.
Feature creation, selection: images principal components **–** calculated the positions of galaxies and stars and then re-centered all of the images.
Other: Sergey has a PhD in physics

Phil Brierley on winning tourism forecasting part two

http://blog.kaggle.com/2011/03/22/phil-brierley-on-winning-tourism-forecasting-part-two/

Model used: no new algorithms created for this competition, rather they took existing algorithms combined their forecasts in specific ways. (Seasonal Naïve, Damped Holt-Winters, ARIMA, and ETS)

Insights: Their mindset was not on how to improve the overall accuracy of a prediction, rather how to prevent worst case events. They did so by not relying on a single algorithm or ensembling.
Feature creation, selection: They removed the last 12 months of training data and used MASE across all the series to assign a weighted average for each algorithm.

I thought it was interesting how they pointed out the problem in the data when reporting a time series on a monthly basis when most operate on a weekly basis. The saw that this caused problems in predicting when for example rental car numbers spike on the weekends, so months with 5 weekends instead of four report spikes in activity.

Other: By looking at the leaderboard they were able to determine the year two growth factor that should be applied. (So there was a little bit of cheating going on – they called It the Cheat factor)

Long live Wikipedia: Dell Zhang on placing third in Wikipedia's Participation Challenge

http://blog.kaggle.com/2011/10/26/long-live-wikipedia-dell-zhang/

Model used: Python was the technology used – Gradient Boosted Trees outperformed all other methods tried.
Insights: His biggest insight was that most users future behavior can be largerly determined based on their recent behavior.
Feature creation, selection: didn't talk much about feature creation but he did mention that parameter tuning was carried out on a big validation dataset shared to all participants by another participant.
Other: Dell Zhang has a PhD on Computer Science

Smile! Alexander D'yakonov on placing third in Photo Quality Prediction competition

http://blog.kaggle.com/2011/11/27/smile-alexander-dyakonov-on-placing-third-in-the-photo-quality-prediction-competition/

Model used: combination of random forests and weighted k-NN – no external information was used.

Insights: His insights in this interview were not hugely helpful, but he did mention that it is necessary to be careful when you build out the final decision. He made a mistake of averaging 3 of his 6 algorithms instead of all.
Feature creation, selection: What helped was taking ratio's of data that was already available. (width of the image divided by the height)
Other: His current profession is an Associate professor

Momchil Georgiev Shares His Chromatic Insight from Don't Get Kicked

http://blog.kaggle.com/2012/02/02/momchil-georgiev-shares-his-chromatic-insight-from-dont-get-kicked/

Model used: unknown
Insights: Do not make assumptions about data – it is useful to generate ideas to jumpstart the analysis process. Through their forecasting they determined that orange cars are much less likely to be lemons than other used cars. The interesting piece of this analysis though is not the outcome rather that you should not read so much into the final outcome.
Feature creation, selection: unknown
Other: Dell

Bond-fire of Data Scientists - Interview with Benchmark Challenge 3rd place Finishers

http://blog.kaggle.com/2012/07/03/bond-fire-of-the-data-scientists-interview-with-benchmark-challenge-3rd-place-finishers/

Model used: 2 blended types of random forests
Insights: They said it was all about how the features were constructed. It was very interesting that the time spent re-implementing random forests in Java manually didn't give much advantage over simple features built in a couple of minutes.
Feature creation, selection: There was no preprocessing done in this example. They in feature selection they used weighted averages over the whole 10 day trade range and the most recent 5 trade range. They also used data normalization to get rid of absolute values.
Other: Profession for this team was made up of an American diplomat and a physicist. Being on a team and combining efforts was one takeaway they found very useful.

Troll Detection with Scikit-Learn

http://blog.kaggle.com/2012/09/26/impermium-andreas-blog/

Model used: He used character n-grams, chi squared and logistic regression. The final submission contained a combination of all of his models. The combination performed better than any single model by itself. The final model that won was the "char_word_model" using sklearn.
Insights: Like other competitions it seemed like the simplest model worked the best. Feature selection and extraction are very important.

Feature creation, selection: he used scikit-learn for feature extraction. L1 features selection w/ logistic regression followed by L2 penalized logistic regression. Using bad work counter he found the most significant features were just choosing those with the most numbers of worlds in the badlist, ratio of words in the badlist, and ratio of words in all caps. He also extended the words in the badlist by taking synonyms in a thesaurus under "stupid"
Other: Intern w/ Microsoft Research

Overkill Analytics: WordPress Winner Describes His Method

http://blog.kaggle.com/2012/09/21/overkill-analytics-wordpress-winner-describes-his-method/

Model used: node centrality (modified version simplified for his needs)
Insights: 90% of the likes have a distance of 3 edges/likes traversed. This is a blog that the subject hasn't immediately liked, but rather in the history of people who have liked the same article as the subject.
Feature creation, selection: node centrality calculation for creating a recommendation engine.
Other: I find it interesting after reading a few of these that most of the winners seem to stay pretty simple with their approach and just try to blindly throw as much at the problem as they can until something sticks. He calls in *overkill analytics* in this blog post.

Team DataRobot: Merck 2nd place Interview

http://blog.kaggle.com/2012/11/04/team-datarobot-merck-2nd-place-interview/

Model used: They used a number of models but the ones that they decided on were Support Vector Machine (SVM) and Gradient Boosting Machine (GBM)
Insights: The two models chosen were very important. The learning they took away from the competition was that it is important to have at least two very different models to improve accuracy. One model may be good for a particular answer but score very bad on another.
Feature creation, selection: unknown
Other: Profession – runs a consultant company specializing in predictive analytics

Tuzzeg the Troll-hunter: Impermium 2nd place Interview

http://blog.kaggle.com/2012/10/18/tuzzeg-the-troll-hunter-impermium-2nd-place-interview/

Model used: logistic regression and random forests
Insights: Again like almost all of the other posts that I looked at the simple model seemed to work best over complex models. By being able to analyzing sentence level features he found that insulting posts were most likely one sentence while longer posts there was just one insulting sentence.
Feature creation, selection: stemming and dependency preprocessing and a language model code that he wrote on his own.
Other: Profession – Worked for a Russian search engine company doing text classification

The interview I found to be the most helpful was with Rouli Nir.

1) The models that appear to be used frequently are random forests and decision trees. Also Bayesian analysis is used a lot by the people winning competitions. Then the modeler uses some other technique to boost their solution from there. The other key that many people mention is their feature engineering and how important that is.
2) Insights:
    a. Trust your k-folds validated model even if it is giving you a worse score on the public dataset.
    b. If you find leakage in a competition use it to your advantage and tell kaggle about it afterwards. ;)
    c. Always use random seeds. You will want to be able to recreate your results
3) Feature selection was pretty much anything goes. Think of a creative way to parse the data and pull out something useful. Be prepared for lots of things to fail, for things that you think will be awesome to fail and things that seemingly have little importance to be a good predictor.
4)
    a. The tutorial on pandas was really helpful. I didn't know about it before this assignment so that was fun to learn.
    b. This is a really good place to find people to follow on twitter and find new data science blogs.

Q&A With Guocong Song, 3rd Prize

Job Salary Prediction Competition

April 23, 2013 by Guocong Song

This was a text mining problem that should usually be done suing classification but was completed with regression by the author. For preprocessing, he had used text normalization, stop words, n-grams, TF-IDF for the text feature extraction. Ridge regression, SGD, random forests were tried on the processed data. He had also tried converting the problem into a classification one with native Bayes, SVM, and logistic regression. The final model utilized was SGD regression and logistic regression based predictor. The tools utliized were python and the scikit-learn library.

The insight into this problem was that salaries were not distributed smoothly. Models that explored local properties would outperform linear regression. There were no significant features that could be capitalized on. Surprisingly, the first and second place finishers utilized neural networks.

Getting Started with Pandas

Predicting SAT Scores for New York City Schools

January17, 2013 by Chris Clark

This post was an introduction to the Panda library for data preparation, joining, and generating tabular data. The data used as an example was the NYC Open Dataset, comprising of five datasets about schools or districts. It was easily ingested since the data was already in csv form. Joining the data from the five datasets was easily done with the join function from the library but they did have to be normalized first. The dataFrames used in the library enabled easy manipulation of the data to shape before joining. Categorical variables were easily dealt with by exploding them into multiple boolean columns.

1st Place: Observing Dark Worlds

December 19, 2012 by Tim Salimans

Bayesian analysis was the winning model in this competition:

1. A prior distribution was constructed for the halo positions that surround dark matter.
2. A probabilistic model was then constructed for the observed ellipticities of the galaxies.
3. Bayes rule was then used to get the posterior distribution of the halo positions.
4. Expected loss with respect to the posterior distribution was then minimized.

The rest of the competition was implementing the Bayes model with respect to the given data. The final test cases that were used to evaluate the model was small, comprised of 90 test cases. It was susceptible to noise. The public leaderboard and the one that was used to evaluate the model did not match up. His model was only above average on the public leaderboard.

Deep Learning How I Did It: Merck

1st place interview

November 1, 2012 by George Dahl

A team comprised of professors and Ph.D. students from University of Toronto and University of Washington won this competition. They wanted to show the effectiveness of neural networks on disparate problems. No feature engineering and minimal preprocessing was done. The model was allowed to learn the features. The final model had single-task neural networks, multi-task neural networks, and a Gaussian process regression. These three models were then averaged equally to produce the final result. Python was used to write their own implementation of the neural nets and scikit.learn was utilized for various utility functions.

The most important insight gained was that the all fifteen tasks could be analyzed equally well as the input to the neural net and the same number for the output layer. Surprisingly, ridge regression for model averaging did not provide any improvement over simple averaging. Gradient boosted decision-tree ensembles were explored at the end since they provided different predictions from the models they were using and increased their averages over weaker individual performances. The key take away is that deep neural nets can outperform other methods on a variety of tasks.

1st place interview for Arabic Writer Identification Challenge

May 3, 2012 by Wayne Zhang

This particular problem is a multi-class classification problem. The provided features were utilized. Linear discriminant analysis was done to win this competition. Matlab was used to implement his own particular algorithm of LDA since it has excellent matrix computation support.

The most important insight was that the training and test set were small. This meant that the model had to be able to generalize across the given problem space. The key take away is that overfitting is a serious problem in the real world. There is always limited training and validation data but the test data is usually unbounded. In general, one problem of Kaggle competitions is that the is a significant discrepancy between public and private scores.

Chucking everything into a Random Forest: Ben Hamner on Winning

The Air Quality Prediction Hackathon

May 1, 2012 by Ben Hamner

To construct the features, the lagging N components were utilized from the full time series, the wining submission was N=8. Targets were gained from each of the 10 prediction times and 39 pollutant measures. The model constructed was 390 Random Forests, one for each predicted offset time-pollutant combination. He utilized Matlab by random choice.

There was no specific insight into the data. It was barely examined before training the model. No domain insight was used to wind the hackathon. The key take away is that Random Forest models are pretty effective on a variety of real-world problems.

Mind Over Market:

The Algo Trading Challenge 4th Place Finishers

January 26, 2012 by Will Cukierski

The team that placed fourth in this competition was a combination of a team of two and a lone wolf participating in the event. They merged their models to better their chances of placing first. The initial models were a kNN model that was fed hand picked feature set and a variety of basic linear regression models fed 30 features from the original data set with weighted rows depending on their appearance on the training and test set. R, specifically with the glmnet package, numpy, and Matlab were utilized to produce the models.

The most important insight into the data was that it was unpredictable during the stock market open. Different models were used to specifically focus on stock movement during the market open. Regression was also done separately for buyers and sellers of a stock since bid and ask prices followed separate paths depending on who initiated the trade. The lessons they learned was that focusing on a more finely tuned model would have increased their standings. Also, market open and outliers had a large impact on the RMSE. The key take away for this competition is that the problem was better tackled as a team.

Owen Zhang on Placing 2nd in the

Claim Prediction Challenge

January 30, 2012 by Owen Zhang

Numerous entries were submitted to find out if the methods used on the validation data would work against the public test set also. He admits that his 3rd serious submission out of the 20 would have placed him 2nd in the competition.

There was a diverse group of modelers and miners competing in this event. The majority of them were machine learners over statisticians. Not that many were from a P&C insurance background.

Picture Perfect: Bo Yang on winning the Photo Quality Prediction competition

Novem ber 23, 2011 by Bo Yang

The dataset was clean and contained no missing values, so it did not require preprocessing. The methods used for the event was a mix of Random Forest, GBM, and two forms of Logistic Regression. Geo-spatial data was added to as features from the Google Maps API and population density data. The helped a little bit.

The was not much insight gained from the data. The winner was surprised that he could not get any signal out of the word pairs from the album's name, description, and photo caption. The key take away from this competition is that it is usually not worthwhile to spend too much time on external data if useful data is already included. His time could have been better spent on perfecting the algorithms and included variables.

Dave Slate on Winning the R Challenge

February 15, 2011 by David Slate

A large number of feature variable selections and parameter settings was used but testing was not systematic enough to conclude that the winning model was optimal. Supplemental data was used from a provided package in the contest. The model used for this contest was “Ensemble Recursive Binary Partitioning”. This algorithm could handle large number of records and features, either boolean, categorical, or numeric. Some synthesized features were generated from crude text analysis. Cross-validation and feedback from the leader board was used to tune the model. Variables were weighted to more highly predictive models.

The core forecasting engine was written in C with a Lua front-end.

Interview: Jeremy Howard on winning the Predict Grant Applications Challenge from the University of Melbourne. This challenge predicted which grant applications would be successful.

What models did they use:
A generalization of random forest was used. He developed his own variation of the random forest algorithm in C# but had not released the code yet.

Insights:
For many fields, a null value was highly predictive. Missing values matter.

Feature creation and selection:
He used Excel pivottables to find groups with low or high grant application success. He then looked at these groups to identify strong predictors. He used C# to normalize data and created Grant objects and Person objects for modeling and he used C# to create features.

He grouped small sets of features into categorical features. He replaced continuous features which had null values with 2 different features – a binary feature which was 1 if the continuous feature was null and another continuous feature which replaced the null values with the median value.

Interview: Keith T Herring (2nd place) on techniques used in the Wikipedia Participation Challenge. This challenge predicted the number of edits an editor will make 5 months after the end date of the training data set.

What models did they use:
Variation of Breiman and Cutler's random forest algorithm. He randomized the random forest model parameters for each individual tree rather than using the same parameters for all trees. He generated 3000 models and then used an optimized ensemble of 34 models as his final predictor.

Insights:
A randomly selected Wikipedia editor had 85% probability of having no new edits in the next 5 months. The most predictive feature (out of 206 features) was an exponentially weighted feature for edit volume of a user.

Feature creation and selection:
He generated his own training set by scraping the edit history of 1 million Wikipedia editors using web scrapers he wrote himself. From this raw data, he created a set of 206 features.

Interview: The Perfect Storm team on winning the Give Me Some Credit competition which predicts the probability that an individual will experience financial distress in the next 2 years.

What models did they use:
They used five supervised learning methods: a random forest of classification trees, a random forest of regression trees, a classification tree boosting algorithm, a regression tree boosting algorithm, and a neural network. The final model was an ensemble of these methods.

Insights:
In this challenge, neural networks outperformed the bagging and boosting algorithms.

Feature creation and selection:
This data set had a limited set of features so there wasn't much processing to do. They identified 2 key predictors – the total number of late days and the difference between income and expense. They used SQL, SAS, R, Viscovery, and Excel.

Other observations:
This was a data mining team consisting of members with complementary skills. Their combined insights help them to win.

Interview: Deep Learning Team on winning Merck's Molecular Activity Challenge. The goal of the Merck Molecular Activity Challenge was to predict the biological activity of different molecules.

What models did they use:
They used single-task neural networks, multi-task neural networks, and Gaussian process regression. They used a weighed average of these model types. They used MatLab code and research code they had developed in python.

Insights:
Deep-learning methods have advanced and can out-perform other machine learning methods.

Feature creation and selection:
They did no feature engineering and minimal pre-processing. For some models, they did log transform some of the features.

Interview: Team DataRobot who was second place in the Merck Molecular Activity Challenge.

What models did they use:
They tried Random Forests, PCA, GBM, KNN, Neural Nets, Elastic Net, GAM, and SVM.

Insights:
Overall, GBM and SVM produced similar results. However, for certain observations GBM outperformed SVM. Domain knowledge and data visualizations did not seem to help.

Feature creation and selection:
This was not mentioned in the interview. They did use R and python for their modeling.

Interview: Team '.' who won third place in the Merck Molecular Activity Challenge.

What models did they use:
They tried different algorithms - SVMs, Random Forest, and neural networks. They used SVMs, gradient increases, and neural networks to build models and then an ensemble of the models was submitted.

Insights:
Pre-processing and modeling large data sets is a challenge. They didn't think that running GBM to select features would work.

Feature creation and selection:
They used GBM on a subset of the training data to select features.
They used SVD to reduce the number of features.

Interview: Xavier Conort who was second place in Give Me Some Credit competition.

What models did they use:
He used a mix of 15 models: Gradient Boosting Machine (GBM), weighted GBMs, Random Forest, balanced Random Forest, GAM, weighted GAM, Support Vector Machine (SVM) and bagged ensemble of SVMs. The best model was an ensemble without SVMs.

Insights:
The likelihood of being late was the most important predictive feature. Using an ensemble of models was important. Ensembling weaker models can be better than an individual strong model. GBMs are powerful.

Feature creation and selection:
He did not do much pre-processing. He did create a new feature which estimated the likelihood of being late by 90 days. This was the most predictive feature.

Interview: Joe Malicki who was third place in Give Me Some Credit competition.

What models did they use:
He used random forests, gradient boosted decision trees, logistic regression, and SVMs. He tried positive/negative feature balancing. The winning model was an average of 50/50 balanced boosting and 10/90 balanced random forest.

Insights:
Use cross-validation on large data sets. Combine the best models for the algorithms you use.

Feature creation and selection:
Because defaults on loans were rare, he used stratified sampling to create his training set. He used domain knowledge to add new features.

Interview: Dell Zhang who placed third in Wikipedia's Participation Challenge.

What models did they use:
Gradient Boosted Trees outperformed the other models he tried.

Insights:
User's future behavior is heavily dependent on their recent behavior.

Feature creation and selection:
He focused on dynamic features. Features reflecting recent behavioral dynamics were strong predictors.

Interview with Idlefons Magrans who won the Algorithmic Trading Challenge

What models did they use:
He tried SVM, GBM, logistic regression, and Random Forests. His final model used Random Forest.

Insights:
The data set was quite variable due to stock market volatility. Different training sets generated different results.

Feature creation and selection:
This was not mentioned in the interview. He used R for pre-processing and modeling.

Job Salary Prediction Link

Interviews 1 2

Goal was to predict the salary of any UK job based on its contents.

This had some easy candidates for feature selection. Location, job type, Full description.

Random forest and ensemble methods look like best initial approaches

The approach that came second was a simple neural network with two hidden variables.

Ad similarity with difference in salary was a hurdle to overcome.

Some Text feature extraction techniques include text normalization, stop words, n-grams TF-IDF

Some other techniques tried ridge regression, SGD, random forests, native Bayes, SVM, and logistic regression

Event Recommendation Engine Challenge Link

Interviews 1

Goal was to predict events the user might be interested in based on users action data.

First look, it is going to be difficult to predict the efficiency of the algorithm or technique.

Used pandas, ipython notebook scikit-learn

Classification problem

Random forest, Gradient Boosting Classifier, naïve bayes

Observing Dark Worlds Link

Interviews 1 2

Goal was to find dark matter halos from data

Bayesian analysis provided the winning solution

Lot of statistical analysis of the location of the dark matter was used to build a Gaussian distribution

Maximum likelihood Estimator was used to Benchmark

Detecting Insults in Social Commentary Link

Interviews 1 2

Goal is to predict the comment posted in a public discussion is insulting.

Logistic regression and random forest for ensemble were used

Simpler text based features worked well for the winning team

Scikit-learn

Problem was solved using some nlp features and ensemble methods

Consumer Products Contests Link

Interview 1

Goal was to match product mentions in free form text description

The winning team used supervised Learning algorithm called “Conditional Random field” after some data pre processing.

Used a tool called MALLET

Lot of heterogeneous data.

The problem involves a lot of language processing to extract product mentions.

Text processing skills helped extracting features. A list of profane words were collected to build a model

GigaOM WordPress Challenge Link

Interviews 1 2

Goal is to predict which people will like which blog posts

Challenges were this is free form text, extracting features was not easy or straightforward.

There is a lot of free form text parsing making python a good choice to pick

Text processing: filter out stop words.

Closest semantic similarity is like calculating distance in k-means clustering.

There is a lot of computational cost involved in cleaning up the data before actual technique can be applied.

Creating a graph out of word press data looked like a good visual representation of data.

Node distance between blog posts sounded like a good idea to estimate similarity and seemed like a good measure of "probability of user liking it"

It looks like it is a general principle to use graph representation for anything, which has a distance so traversing is easier.

Stay Alert ford Challenge. Link

Interviews 1 2 3

Goal is to predict Driver Alertness.
Preprocessing of data was necessary to remove some noise from data
Some techniques used were Support vector machine (SVM) using libsvm, python
Excel sqlserver Decision tress and Neural Network
What works best on training set did not work well on test set

Wise words

- Be careful with feature variable selection. All variables might not be needed
- Watch out for over fitting. Reducing Model Complexity Yields better prediction on real data
- In some situation Domain knowledge is as important as data skills

Some Algorithm and technique names dropped

Ensemble Recursive Binary Partitioning
Bagging (Bootstrap Aggregation)
Boosting
AdaBoost(Adaptive Boosting)
Linear Regression
Classifiers
Decision Tress
k-NN Nearest Neighbours
Random Forests
CART (Classification & Regression Tree)
Ensemble Methods

Tools of Trade

R
Python, pandas, numpy, scipy and scikit-learn
Ipython notebook
Matlab

More Tools Suggested for Data Scientists

1. Angular.js for building web apps
2. Makefiles

“5 Lessons Learned from the Event Recommendation Challenge”

Model is based on Random Forest and Gradient Booster classifier. Insights were that a new model wasn’t needed to win 3rd place, the important part is working with the data. A major lesson learned is that Random Forests can still overfit, as he added features the score went down. It was critical for him to derive some features, like the location a person is in relation to an event.

Troll Detection with Scikit-Learn

Winners 1, 2, 4, 5, and 6 all used scikit-learn, an open source machine learning library in Python. The solution for #6 winner was to use n-grams, handcrafted features, chi squared and logistic regression.

Overkill Analytics: WordPress Winner Describes His Method

He wrote ‘ugly Python script’ to create the features and then threw it into a glm and a Random Forest. He says a key to all the winners was using a little bit of the Wordpress social graph. A node centrality calculation was a key to this winner’s success.

The Dangers of Overfitting or How to Drop 50 spots in 1 minute

This wasn’t actually a winner, but someone who dropped substantially because they had over fitted the data used on the public leader board, which gave a great score, but when the competition was finished and everyone is scored on the full test set, the over fitting becomes apparent. This could have been prevented by doing better cross validation.

1st place interview for Boehringer Ingelheim Biological Response

They used Random Forests for feature ranking and selection. “Simple transformations and splines were used for some models” (not sure what this means, ‘splines’ as in linear regression splines?)

Bond-fire of the Data Scientists - Interview with Benchmark Challenge 3rd place Finishers

Used two different Random Forest models, blended using stacked generalization. Most features were created manually. A key insight was that data normalization helped build a stronger model. Also, sometimes some features that take a couple of minutes can do much more than harder things like re-implementing Random Forests.

1st place interview for Arabic Writer Identification Challenge

Used all the provided features. Used Linear Discriminant Analysis. Used Matlab because of superior Matrix computation support. Used a tool called LDA which used to be popular for face recognition.

Grockit 2nd place interview with Alexander D'yakonov

Approach was to reduce problem to standard classification problem. He added linear combinations of features for performance. He blended GBMs from R, GLM from Matlab, and Neural Nets from CLOP library in Matlab. He was surprised Random Forests performed worse than GBMs.

Vladimir Nikulin on taking 2nd prize in Don't Get Kicked

Neural Nets and GBM in R. The Neural Nets were only used to calculate the coefficients of the weighting in the blending model. A key insight into the data is that the relation between prices were more interesting than the prices themselves.

The Perfect Storm: Meet the Winners of 'Give Me Some Credit'

They used 5 supervised learning methods: a Random Forest of Classification trees, a random forest of regression trees, a classification tree boosting algorithm, a regression tree boosting algorithm, and a neural network. Their major insight was that there were 2 key features that had the greatest predictive power.

1st Place: Observing Dark Worlds

December 19, 2012

Tim Salimans

1. Models: probabilistic model
2. Insights: Keeping the model simple kept it from overfitting the data.
3. Features: A decreasing function in the Euclidean distance r_{ij} between galaxy *i* and halo *j*
4. Observations: A simple random-walk Metropolis Hastings sampler was used to approximate the posterior distribution of the halo distributions

Deep Learning How I Did It: Merck 1st place interview

November 1, 2012

George Dahl

1. Models: Supervised learning: single-task neural networks, multi-task neural networks, and Gaussian process regression.
2. Insights: Their architecture allows the network to reuse features it has learned in multiple tasks and share statistical strength between tasks. Exploring more feature engineering and preprocessing possibilities might have given a better solution.
3. Features: For some models, they log-transformed each individual input feature/covariate. However, they prefer to learn features rather than engineer them.
4. Observations: Deep learning can have excellent results given enough time and experimentation.

How We Did It: CPROD 1st place interview

October 4, 2012

Sen Wu

1. Models: Conditional Random Field, CRF + Standard and Rule Template
2. Insights: Combinations of CRF models achieved the highest score in the private leaderboard but not in the public leaderboard.
3. Features: characters of products names and human semantic behavior analysis
4. Observations: Semantics used in the forum environment poses a challenge.

Overkill Analytics: WordPress Winner Describes His Method

September 21, 2012

Carter

1. Models: Putting calculated features into a glm and a random forest, then averaging the results.

2. Insights: Finding the key common readers and common blogs with a graph is a promising avenue for finding recommendation candidates that are new to the subject user.
3. Features: Node centrality calculation
4. Observations: The 'like graph' is a more practical avenue for a recommendation engine. Finding *unseen* blog posts in which a user may have interest would be a more relevant and valuable tool. Finding new blogs for the user with high centrality in their social graph is a reasonable way to find them.

1st place interview for Arabic Writer Identification Challenge

May 3, 2012

Wayne Zhang

1. Models: LDA (popular and successful in face recognition)
2. Insights: Since the data sizes were small, he had to be careful about the generalization ability of his model.
3. Features: A linear discriminant analysis is suitable for a multi-class classification problem such as this.
4. Observations: One challenge of Kaggle competitions is the discrepancy between the public and private scores.

Of Caffeine and Cross Validation: Tim Veitch on Don't Get Kicked!

February 1, 2012

Tim Veitch

1. Models: GBMs, random forests, logit model (building logit trees with a logistic regression on each leaf)
2. Insights: the temporal effects suggested that an eight month period were the months with the lowest 'kick likelihood'.
3. Features: regression: wheel type, month, lack of change in the MMR price
4. Observations: Kaggle has really reinforced the importance of cross validation. And drink lots of coffee… but not too much!

Speaking in Hands: Winner of Round 1 of the CHALEARN Kinect Gesture

May 9, 2012

Alfonso Nieto Castanon

1. Models: a bayesian network model for recognition
2. Insights: a system that would mimic the specificities of the human visual system would be most likely to pick up on the helpful cues from the video sequences correctly.
3. Features: ad hoc features from the depth videos
4. Observations: He developed problem-specific algorithms rather than using a combination of off-the-shelf procedures.

The Perfect Storm: Meet the Winners of 'Give Me Some Credit'

January 3, 2012

Daniel McNamara

1. Models: derived from expertise in credit scoring and algorithms

2. Insights: two key features were very predictive, the total number of late days and the difference between income and expense.
3. Features: a random forest of classification trees, a random forest of regression trees, a classification tree boosting algorithm, a regression tree boosting algorithm, and a neural network.
4. Observations: Working in a team they had expertise in a variety of areas which added up to a winning team when otherwise they would have been unlikely to win as individuals.

Kernel Density at the checkout: D'yakonov Alexander on winning the dunnhumby Shopper Challenge

October 16, 2011

Dyakonov Alexander

1. Models: probabilities of visits and also the stability of users' behavior
2. Insights: it was necessary to take account of the fact that 'fresh' data is more useful than 'old' data, so he used a weighted Parzen scheme to give greater weight to more recent data points.
3. Features:heuristics – breaking into two parts, the date and dollar. Unable to predict date before the dollar.
4. Observations: He only used MATLAB and just used the basic M-language without any libraries.

From Soundtracks & Signatures to Stars & Galaxies: Ali Hassaine and Eu Jin Lok

October 4, 2011

Ali Hassaine

1. Models: the morphological approach (methods inspired from soundtrack restoration)
2. Insights: The soundtrack film is sometimes badly exposed due to light diffusion during the copying process. This concept relates to the exposure elements involved in the images of stars and galaxies.
3. Features: methods inspired from signature verification and writer identification (directions, curvatures, edge distribution, chain codes)
4. Observations: Several geometrical features previously used in signature verification and writer identification happen also to be very powerful predictors for computing the ellipticity.

What model(s) did they use

Having read more than 20 interviews, I would consider splitting the winning models for final submission from the ones that were tried to test different solutions.

To a large extent, the following three models appear in final submission: Neural Network, Gradient Boosting Machine and Random Forest. The two latters appearing often together as ensemble in final submission. To a lower but still significant extent: Support Vector Machine.

Insights

- It is often reported that a lot of time is spent on understanding and cleansing the datasets
- Features engineering seems to be key
- Domain knowledge is important. Often, participants spent a lot of time reading information to understand the domain since it eventually helps for features engineering
- It helps to simply try out models. Some competitors reported their good surprise (e.g. Gradient Boosting Method for the Merck challenge) or disappointment (Naïve Bayes for the job salary

prediction competition or SVM for the Merck challenge) about some models. However, in many cases final submissions were made of several models.

Feature creation, selection

I only found one single case where a team did not do Features Engineering on purpose because they also wanted to prove that Neural Network method does not need it. They won the 1st place of the Merck competition. Otherwise, in all other interviews I read, Features Engineering (generation, extraction, creation, selection) was considered the key to success.
Having read the paper "*An Introduction to Variable and Feature Selection*", I wonder how far Features are engineered in that way. Admittedly, there is no in depth information on that topic in the interviews but it seems to me that preprocessing is often done with models e.g. Neural Network or SVM based classifiers that already embed selection/discrimination of features. Also, "brute force", or trials and errors, seems to be a usual approach for selecting features e.g. the winners of the IJCNN Social Network Challenge.

Other observations

- R and Python are the most widely used set of tools
- Teams are often made of Computer Scientists (or IT engineers) and Statisticians. People competing alone have more often Computer Science (or IT/Software engineering) background.
- Winners have usually taken more than one Kaggle challenges before being in the top three

1) What model(s) did they use?

Common ones Are neural network, random forest, GBM

2) Insights?

Let the data speak to you seems to be a common insight. While some subject matter expertise can help in certain situation, at least in one instance, its seems to have made the model worst. Some times, the data set can be too big to fit into RAM, So require special handling.

3) Feature creation, selection?

Feature selection Seems to Be the key. Over fitting seems to be a commonly mentioned problem.

4) Other observations?

While there are a lot of tools out there: R, Matlab, Python, Perl, C, C++. The Common ones seems to be R, Matlab, and Python.

1- Event Recommendation

The goal is to predict events users will be interested in based on events they've responded to in the past. The winner, Rouli Nir (#3), used the following tools: pandas, sci-kit-learn and iPython notebook.
Model(s) : Random Forest, Gradient Boosting Classifier, Naive Bayes
Insights : Comparing the delta of each event displayed to a user against the delta of the earliest event proves to have more predictive power than the number of friends a user had in the event.
Feature creation, selection: delta(the time between when the user was presented with the event and the event's date), the ratio of invited users that decided not to attend an event, the total number of attendees, and the estimated distance between the user and the event.
Other observations – winner's take-aways are lessons about handling data.

2- Deep Learning

The goal is to show the effectiveness of neural networks. The 1st place winners are part of a team from the University of Toronto. They used the following tools: Matlab code, gnumpy , Python, Scikits.learn
Model(s) : single-task neural networks, multi-task neural networks, and Gaussian process regression.
Insights : Using all inputs from all tasks with different output units allows the network to reuse features it has learned in multiple tasks and share statistical strength between tasks.
Feature creation, selection: no feature engineering, learn features rather than engineer them.
Other observations – take-aways for winner is that training procedures for deep neural networks have now reached a stage where they can outperform other methods on a variety of tasks, not just speech and vision.

3-Troll Detection

The goal is to classify forum posts / comments into "insult" and "not insult". The winner, Andreas Mueller(#6) used the following tools: Python, Scikits.learn
Model(s) : logistic regression
Insights : the simplest model worked best
Feature creation, selection: use features from PCA and K-Means (distance to centers).
Other observations – take-aways for winner is feature selection and feature extraction are very important.

4- Boehringer Ingelheim Biological Response

The goal is to predict a biological response of molecules from their chemical properties.
The winner, Jeremy Achin(#1) used the following tools: R & Matlab
Model(s) : Random Forests, GBM, KNN, Neural Nets, Elastic Net, GLM, GAM, and SVM
Insights : Eliminating variables reduced model training time, and improved performance considerably
Feature creation, selection: Random Forests for feature ranking & selection
Other observations – take-aways for winner is having a team with diversity and relentless tenacity is extremely important.

5- Don't get Kicked

The goal is to predict if a car purchased at auction is a lemon. *Tim Veitch, the 4th prize winner* used the following tools: C++, R, Ruby, and Excel
Model(s) : Random Forests, and GBM
Insights : the variables which proved important: wheel type, the month, or a lack of change in the MMR price .
Feature creation, selection: ordinal variables from each of the numeric variables
Other observations – It was surprising how relatively unimportant the make, model and vehicle type were.

6- Algo Trading

The goal is to predict the market response to large trades. *TIldefons Magran*, THE WINNER, used R.
Model(s) : Random Forests
Insights : stock market conditions are extremely volatile.Different samples of the training set could fit very different models.
Feature creation, selection: SVM, LR, GBM, RF
Other observations – Winner had to improve parallel programming skills in R.

7- Mapping Dark Matter

The goal is to measure the small distortion in galaxy images caused by dark matter. Team *DeepZot(#1) uses the following tools : C++ and* C++ libraries.
Model(s) : Neural network

Insights : the pixel-level residuals are a powerful tool for finding the best-fit models for galaxy and star images
Feature creation, selection: use of the maximum amount of information available in the images in the presence of pixelation and noise
Other observations – Without the neural network, our best entry would have ranked 8th

8-Tourism forecast, part 1

The goal is to forecast tourism. The Team "Sali Mali" which won the seasonal part used R packages
Model(s) : Damped Holt-Winters, ARIMA, ETS
Insights : It was found that just multiplying the 1st year's predictions by approximately 1.04 gave the best second year predictions.
Feature creation, selection: tourism figures are likely to be affected by global factors (GFC, exchange rates, etc., that are largely unpredictable) .
Other observations – the conclusion that one algorithm is better for tourism data than another algorithm must be treated with caution as it might just be the algorithm that fortunately got the trend correct for that particular moment in time.

9 - Grant Applications

The goal is to predict Grant Applications. Jeremy Howard, a Kaggle employee, used the following tools : Excel pivot tables, C#
Model (s): random forest
Insights: My approach to this competition was to first analyze the data in Excel pivot tables. I looked for groups which had high or low application success rates. In this way, I found a large number of strong predictors
Feature creation, selection: grouping up small groups in categorical variables, and replacing continuous columns with null values with 2 columns (one containing a binary predictor that is true only where the continuous column is null, the other containing the original column, with nulls replaced by the median).
Other observations – used C# to normalize the data into Grants and Persons objects, and constructed a dataset for modeling.

10 -Tourism forecasting, part 2

Jeremy Howard explained that he used the following tools: C#, in Visual Studio 2010
Model(s): weighted regression
Insights: A simple optimizer found the optimal weighting of each in order to predict the final 2 years. This weighted model was then applied to the full data set to create predictions.
Feature creation, selection: automatically remove outliers, customized local smoother,and used the residuals to determine seasonality
Other observations – Winner spent a couple of weeks on the problem, including reading and research