Winning @ WiDS Kaggle Datathon

Dan Ofer

Datathon In Numbers

1479 Competitors

951 Teams

2 Months work

12,644 Entries

1 **Winner**

Women in Data Science (WiDS)

Aims to inspire and educate data scientists worldwide, regardless of gender, and to support women in the field.

The Datathon challenge is an annual, international competition, hosted on Kaggle
>= 50% Females per team

Women in Data Science (WiDS) Datathon 2020

OBJECTIVES:

- ✔ Predict patient survival/death in hospital intensive care units (ICU) from their first 24 hours medical records
- ✔ More than 130,000 hospital Intensive Care Unit (ICU) visits from patients, from > 200 hospitals in different countries
- ✔ Also a research paper tract!

OUR TEAM - *WOMAN POWER*

Noa Dagan

MD - Head of AI Driven Medicine
at Clalit (HMO) Research Institute

Nurit Cohen Inger

Data scientist

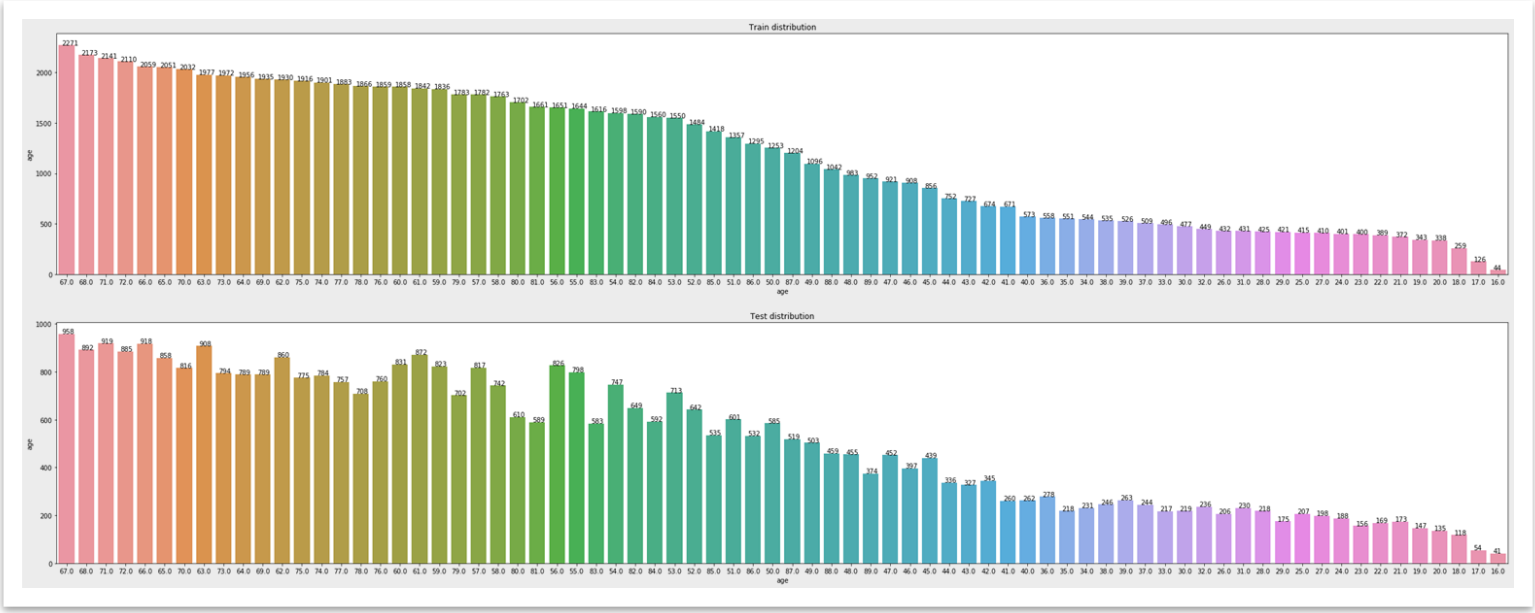
Dan Ofer

SparkBeyond senior DS,
Bioinformatician, led previous
healthcare/AI projects

Seffi Cohen

Kaggle Master
BGU PhD student

The Dataset



THE DATASET

- **Very “flat” data** - no time series, repeat visits, real world entities or free text
- 130,000 samples, 186 columns
- **LOTS of missing variables**, many missing not at random.
- Heterogenous mix of continuous, categorical, semi structured (medical codes)
- **Many domain-specific**, expert selected top features - e.g. APACHE risk scores, aggregated lab tests results etc’

training_v2.csv (61.75 MB) 20 of 100 columns Views

hospital...	# hospital...	A age	A bmi	# elective...	A ethnicity	A gender	A height	A hospital...	A icu_ad...
118	0	68	22.73	0	Caucasian	M	180.3	Floor	Floor
81	0	77	27.42	0	Caucasian	F	160	Floor	Floor
118	0	25	31.95	0	Caucasian	F	172.7	Emergency Department	Accident & Emergency
118	0	81	22.64	1	Caucasian	F	165.1	Operating Room	Operating Room / Recovery
33	0	19	NA	0	Caucasian	M	188		Accident & Emergency

A TOUGH BASELINE

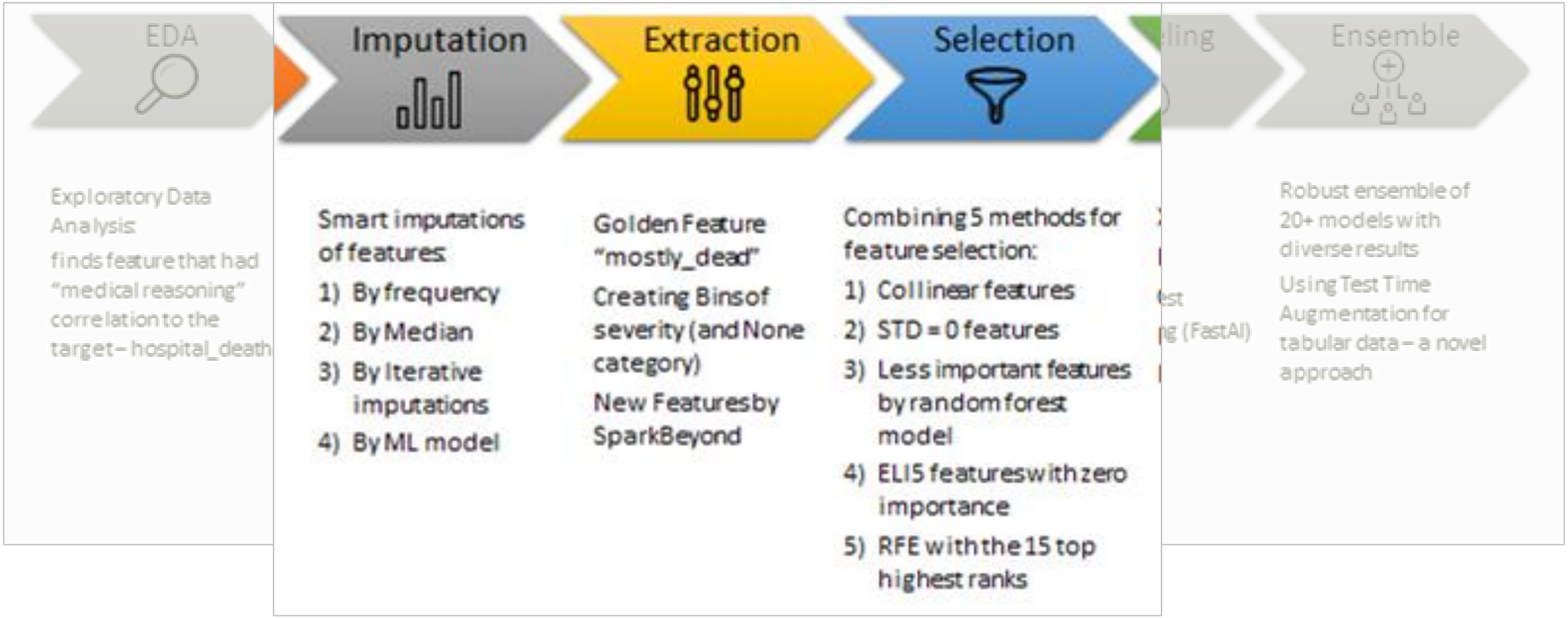
- APACHE death prob feature as baseline: 85 AUC
- [My baseline gradient boosting (Catboost) kernel]() got ~ 90.2 AUC
- A SparkBeyond (~AutoML) model with expert tricks and xgboost got ~ 90.4 AUC
- Our winning model ensemble: 91.497 AUC
- Tiny spread! And strong baseline features!

#	Δpub	Team Name	Notebook	Team Members	Score	Entries
1	▲1	Women Power			0.91497	205
2	▲15	nullset			0.91485	56
3	▼2	Prevision.io			0.91445	216
4	▼1	ML Keksika			0.91431	36
5	▲1	EPFL			0.91407	159
6	▲7	Dihydrogen Oxide			0.91395	123
7	▲8	Polo5			0.91348	80
8	▲3	Team Blue			0.91343	173
9	▼1	Man in Data Science			0.91341	44

OUR PIPELINE

1st place writeup: https://www.kaggle.com/c/widsdatathon2020/discussion/133189

FEATURE ENGINEERING, SELECTION, IMPUTATION



MISSING VALUES IMPUTATION

Many "**missing not at random**" variables -
NOT doing a lab test can be more informative than the test itself!

Default Catboost/tree models worked very well (built-in support for missing values)

- BUT! Imputing missing values by domain knowledge (e.g. BMI: Body-mass index = ht/wt^2) or predictive models - important and improved final model.
- Other models can't work with missing values - necessitating imputation

Very laborious work - 186 columns!

- Clear boost to model performance.
- Outperformed default model imputations

Hard to say if worthwhile in the "real world"...
Except for easily imputable demographic variables such as BMI, age.. which ARE worthwhile!

FEATURE ENGINEERING: **1.** Domain knowledge

Golden feature: should_be_dead

Grouped insights from different biological measurements that are not compatible with life

= Benefits of having a Md. on the team ;)

FEATURE ENGINEERING: **1.** Domain knowledge

- ✔ Many aggregated features - e.g. counting how many multiple chronic conditions or subgroups of chronic conditions a patient had, the total number of tests performed, summation of tests or diagnostics with missing values...
- ✔ Bucketing continuous variables into discrete bins, based on medical knowledge
- ✔ Grouping of related measures to a final score (e.g. Glasgow Coma Scale)
- ✔ Computing kidney function based on relevant lab test and diagnostic codes
- ✔ Computing pulse pressure variables by subtracting the systolic and diastolic Blood Pressure measures
- ✔ Deltas/Diff Variables for a variable per hour +- per day
- ✔ Grouping medical diagnoses codes into semantic categories based on the hierarchical nature of the medical coding system
  - ✔ The codes were semistructured, similar to ICD codes.
  - ✔ e.g. 123 = kidney cancer, 12 = cancer, etc’

And many many more!

FEATURE ENGINEERING: **2.** Universal tricks

- ✔ Row level (or columns' subset) aggregations: count nans, zeroes, max/min
- ✔ Count transformation (replace value with the # times it appeared)
- ✔ Measures of anomalousness - Isolation forest derived anomaly score
- ✔ Hash on subset of columns to create similar profiles (e.g. binned age, weight, gender, lifelong chronic diseases)
- ✔ Add aggregate features by these hashes. e.g. mean risk of death for age+weight profile
- ✔ Feature interactions - restricted to subset of columns and functions (×,÷,max, min, value range/diff).
- ✔ Used SparkBeyond feature selection for this initially, to generate candidates

  e.g. "max Blood Pressure in first hour divided by max BP over first day"

FEATURE ENGINEERING: **3.** SB augmented tricks

- ✔ Not actually used, but “cool”
- ✔ I extracted from an external PDF lexicon descriptions of the APACHE medical codes diagnosis. I then ran SB/world knowledge to get semantic concepts
- ✔ Features were insightful but did not improve model, presumably due to low cardinality (1k<) of the structured codes
- ✔ Concepts associated with **higher risk of death** included: intensive care medicine, **respiration & lungs, infectious diseases** (e.g. sepsis, **infections**)
- ✔ Concepts with a **lower** risk of death included: coronary artery bypass (e.g. open heart surgery), medical specialties (e.g. surgeons), **drugs** (e.g. overdose or drug withdrawals) & **crime**

FEATURE SELECTION (FS)

We used multiple feature selection techniques. We checked these techniques separately and experimented whilst iteratively dropping features in a fast LightGBM pipeline, according to what improved the local validation or Leaderboard score.

- Shapley values helped for high level interpretation

We dropped features using a mix of the following methods:

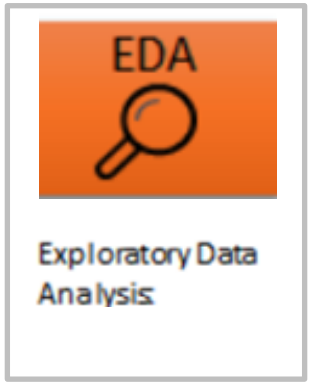
- Features with a high percentage of missing values (80% and up)
- Collinear (highly correlated) features - with threshold above 0.99
- Feature with zero standard deviation (contains just 1 unique value - ‘readmission_status’)
- Features ranked by Recursive Feature Elimination with a linear regression model (e.g. 'paco2_for_ph_apache')
- Features with zero importance according to permutation FS in ELI5 (e.g. 'hepatic_failure')
- We dropped our **top features**, the existing APACHE death risk scores ('apache_4a_hospital_death_prob', 'apache_4a_icu_death_prob', and derived features). We found this improved our validation score, possibly due to bias in how the score is calculated by humans, leading to model overfitting
- Features with a different train - test distribution, or no test set coverage - according to **adversarial validation** models (‘icu_id’, ‘hospital_id’..)

ADVERSARIAL VALIDATION

Set train/test split as target label

Useful for:

- Feature selection
- Identifying **leaks**
- Concept drift
- Datapoint selection:
    - From train set, select points most like test set and use those as local validation set



SparkBeyond was good for this! - rapid iterations

Statistical (K-S) approach didn't work as well - too easy to distinguish train from test

Based on this - we identified that train/test was mostly partitioned on hospital wards.
We dropped related columns (which had been top features), as they **wouldn't generalize to the test** set

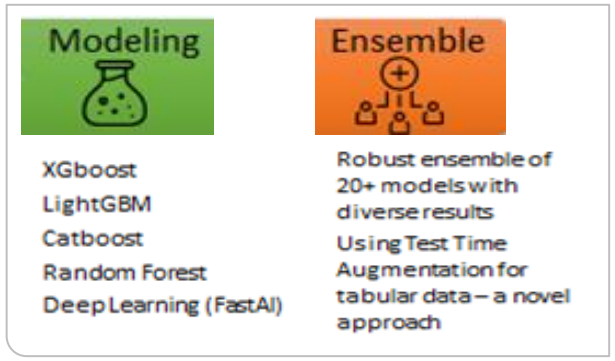
FEATURES

Rank	Feature	Direction of Effect	Score	Train RIG	Train Support	Feature Missing Values	Histogram
1	gcs_motor_apache < 5.759	x2.43 for hospital_death: 1	0.081	0.081	11737 (23%)		
2	d1_lactate_min_div_diasbp_min ≥ 0.066	x5.07 for hospital_death: 1	0.059	0.080	2518 (5%)	74.83% (78.21% of ...	
3	bun_apache ≥ 26.915	x1.79 for hospital_death: 1	0.035	0.049	17557 (34%)		
4	d1_sysbp_min < 80.5	x2.41 for hospital_death: 1	0.034	0.065	10274 (20%)	0.15% (0.80% of True)	
5	d1_arterial_ph_min < 7.242	x4.06 for hospital_death: 1	0.030	0.065	3228 (6%)	66.03% (70.37% of ...	
6	ventilated_apache ≥ 0.5	x2.05 for hospital_death: 1	0.029	0.078	16562 (32%)		
7	left_average_spo2 < 92.5	x3.36 for hospital_death: 1	0.028	0.046	3607 (7%)	0.28% (4.18% of True)	
8	apache_2_diagnosis inRange (112.5 to 115.5)	x2.47 for hospital_death: 1	0.027	0.050	7973 (15%)		
9	should_be_dead ≥ 1	x6.73 for hospital_death: 1	0.027	0.061	1133 (2%)		
10	d1_temp_min < 35.685	x2.55 for hospital_death: 1	0.027	0.044	6661 (13%)	2.55% (20.66% of True)	

FEATURES

Rank	↓ ..	Feature	Direction of Effect	Score	Train RIG	Train Support	Feature Missing	Histogram
9	★	should_be_dead ≥ 1	x6.73 for hospital_death: 1	0.027	0.061	1133 (2%)		
16	★	diasbp_invasive_started_after_firstHour	x2.95 for hospital_death: 1	0.017	0.036	3879 (8%)		
18	★	not isNaN(arterial_po2_d1_value_range)	x1.87 for hospital_death: 1	0.017	0.064	18240 (35%)		
19	★	min(d1_mbp_min, d1_mbp_invasive_min) < 46.5	x3.69 for hospital_death: 1	0.017	0.042	2685 (5%)	74.18%...	
20	★	row_zero_count ≥ 35	x1.05 for hospital_death: 0	0.017	0.054	25328 (49%)		
22	★	fio2_cat = "high"	x3.37 for hospital_death: 1	0.016	0.045	3455 (7%)		
26	★	d1_temp_max notInRange (36.61 to 38.02)	x1.75 for hospital_death: 1	0.013	0.028	12936 (25%)	2.55% (...	
47	★	high_lectate	x8.81 for hospital_death: 1	0.009	0.045	496 (0.963%)		
127	★	ph_cat = "low"	x8.76 for hospital_death: 1	0.002	0.007	84 (0.163%)		
246	★	low_heartrate2	x5.78 for hospital_death: 1	0.001	0.023	585 (1%)		

Models



Modeling

- Gradient Boosting trees (Catboost, Xgboost, LightGBM, H2O GBM)
- Scikit learn (KNN, Logistic Regression, Random Forest, Extremely Randomized Trees, Histogram-Based Gradient Boosting)
- Deep learning models (Pytorch and Tensorflow) - weaker results (~88 AUC) but diverse and contributed to the final ensemble

Ensembled with **Pystacknet** (Python stacknet port) = Automated stacked, boosting, CrossValidated ensembles

Pseudolabelling

Take high confidence predictions on test set, and add them to training set

Ensembling methods

- Diverse data sets - imputation techniques, features, and data generation
- Stacking the models using 3 layers stacknet.
- Normalized the models probabilities.
- Test Time Augmentation (TTA)

TTA is a novel technique that aims to increase accuracy and decrease unfair bias. It works by by generating new similar test sets with mild changes. For example, the age, gender, and ethnic values. Then we ensemble them together.

FINALLY… WE WON!

#	Δpub	Team Name	Notebook	Team Members	Score	Entries
1	▲ 1	Women Power			0.91497	205
2	▲ 15	nullset			0.91485	56
3	▼ 2	Prevision.io			0.91445	216
4	▼ 1	ML Keksika			0.91431	36
5	▲ 1	EPFL			0.91407	159

Results on the private leaderboard:
https://www.kaggle.com/c/widsdatathon2020/leaderboard

Detail writeup of our approach:
https://www.kaggle.com/c/widsdatathon2020/discussion/133189

INSIGHTS

- Tabular data in a complex domain - important to get *deep* involvement of a domain expert, e.g. - a medical doctor
    - Also helpful during data cleaning & imputation
- Ensemble methods always work - Double entendre: “modelwise” and “peoplewise”/”featurewise”
- The existing APACHE risk scores underperform compared to ML models trained on the data, with said models generalizing surprisingly well across to different hospitals/ICU wards
- Even with a relatively large and clean dataset, manual feature engineering is valuable
- Winning a kaggle competition depends on the right team and accurate teamwork, along with a healthy dose of hard work and luck :)
- Our 1st place detailed write up: https://www.kaggle.com/c/widsdatathon2020/discussion/133189

THANK YOU!

Questions