Criteo Click-through Rate Prediction

W261 Final Presentation: Team 2

Danielle Adler, Conor Healy, Craig Fujii, YoungKoung Kim

- **Which machine learning algorithm produce the best predictions of CTR?**

- **What metric(s) should we consider when we are defining "best" predictions of CTR?**

- **How can we implement scalable machine learning algorithms that efficiently handle a large amount of data?**

We chose F1-score as our primary measure because...

- **Takes precision (ratio of the true positives to true and false positives) into account**
- **Takes recall (ratio of the true positives to true positives and false negatives)**
- **Works very well when classes are not balanced**

Logistic Regression – main algorithm of choice: probability that a particular outcome is a success divided by the probability that it is a failure.

example #	y	$t = \beta_0 + \sum_{i=1}^{m} \beta_i x_i$	$p(success) = \frac{1}{1 + e^{-(t)}}$	$\alpha \times (y - prediction) \times prediction \times (1 - prediction)$	β_0'	β_1'	β_2'	β_3'	β_4'
1	0	$1 + (1 \times 0) + (1 \times 0) + (1 \times 1) + (1 \times 0) = 2$	0.88	-0.05	0.95	1	1	0.95	1
2	0	$0.95 + (1 \times 1) + (1 \times 0) + (0.95 \times 1) + (1 \times 0) = 2.91$	0.95	-0.02	0.93	0.98	1	0.93	1
3	1	$0.93 + (0.98 \times 0) + (1 \times 0.18) + (0.93 \times 0) + (1 \times 1) = 2.11$	0.89	0.01	0.94	0.98	1	0.93	1.01
4	1	$0.94 + (0.98 \times 0) + (1 \times 1) + (0.93 \times 1) + (1.01 \times 0) = 2.87$	0.95	0	0.94	0.98	1	0.93	1.01

Decision Trees – secondary algorithm of choice: divides the data in homogenous subsets using binary recursive partitions.

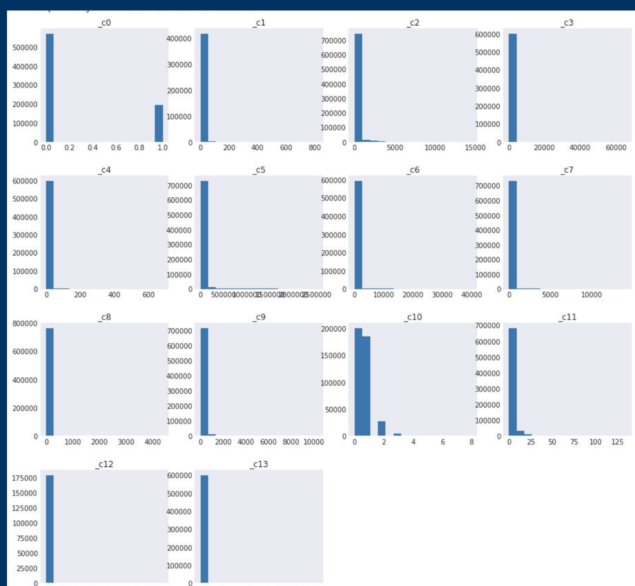
Berkeley

Section 3 | EXPLORATORY DATA ANALYSIS

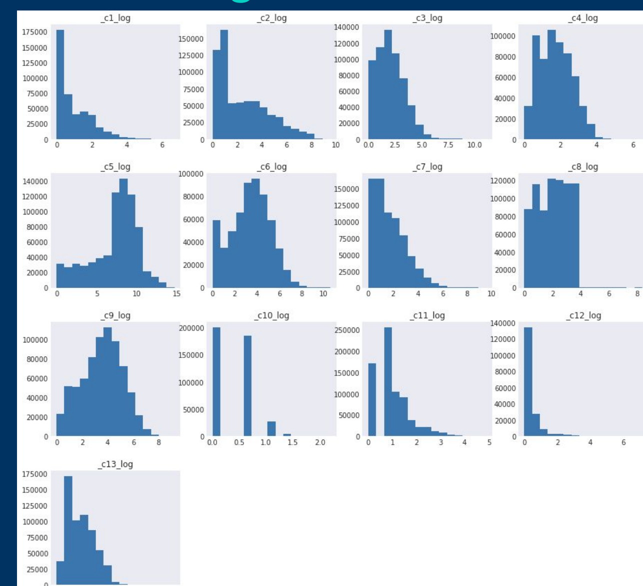
Numeric Variable Insights:

- Heavily right-skewed without log transformations
- Wide dispersion of values; not on the same scale
- Several variables have over 40% null values

Non-Transformed



Log-Transformed



Categorical Variable Insights:

- Categories have a wide dispersion of unique counts
- Many categories have over 30% null values

Distinct Counts per Category:

_c14	_c15	_c16	_c17	_c18	_c19	_c20	_c21	_c22	_c23	_c24	_c25	_c26
1188	549	284917	115407	259	16	10951	523	3	28805	4828	254427	3145

_c27	_c28	_c29	_c30	_c31	_c32	_c33	_c34	_c35	_c36	_c37	_c38	_c39
27	8985	196549	10	3899	1785	3	230184	13	14	38695	68	28259

Proportion of Nulls of Each Category:

_c14	_c15	_c16	_c17	_c18	_c19	_c20	_c21	_c22	_c23	_c24	_c25	_c26
0.0	0.0	0.034	0.034	0.0	0.121	0.0	0.0	0.0	0.0	0.0	0.034	0.0

_c27	_c28	_c29	_c30	_c31	_c32	_c33	_c34	_c35	_c36	_c37	_c38	_c39
0.0	0.0	0.034	0.0	0.0	0.441	0.441	0.034	0.762	0.0	0.034	0.441	0.441

- Six of the categories have their first two features add up to over ~65% of all values, examples shown in __c35:

_c35	count	total	percent
null	581689	763440	0.7619
ad3062eb	104433	763440	0.1368
c9d4222a	64207	763440	0.0841
78e2e389	5592	763440	0.0073
8ec974f4	4007	763440	0.0052
c0061c6d	3045	763440	0.004
ccfd4002	207	763440	3.0E-4
8651fddb	167	763440	2.0E-4
49e825c5	76	763440	1.0E-4
28f45308	9	763440	0.0
d9ce1838	3	763440	0.0
2ec53c35	3	763440	0.0
1856e93d	1	763440	0.0
24eb7cbf	1	763440	0.0

Berkeley

Numeric Variable Transformation:

- **Normalizing**
  - Took the log of all numeric variables and normalized them
  - Created a vector of all numeric features to save computational costs
- **Imputing the mean:**
  - Took the average of the non-null values in the dataset and imputed the mean for the null values
- **Imputing zero:**
  - Replaced all null values with zero

*Written as part of Section 2 Algorithm Theory in the Jupyter Notebook

Berkeley

Categorical Variable Transformation:

- **Binning:**
  - Took the weighted average of the dependent variable for each value in the category
  - Used two binning methods: high / low (including nulls) and high / medium / low / null
- **One-hot encoded our bins**
  - Performed this transformation on the binned categories to avoid over a million columns
- **Weighted average:**
  - Converted the variable from categorical to numeric
  - Imputed the mean for any null values

*Written as part of Section 2 Algorithm Theory in the Jupyter Notebook

Baseline Models:

- Predicting the majority class (F1 score: 0.853)
- Predicting a random class (F1 score: 0.598)
- Predicting a random weighted class of 75% on majority and 25% on minority (F1 score: 0.744)

Algorithm Models (run on a split of a train / test 1M dataset:

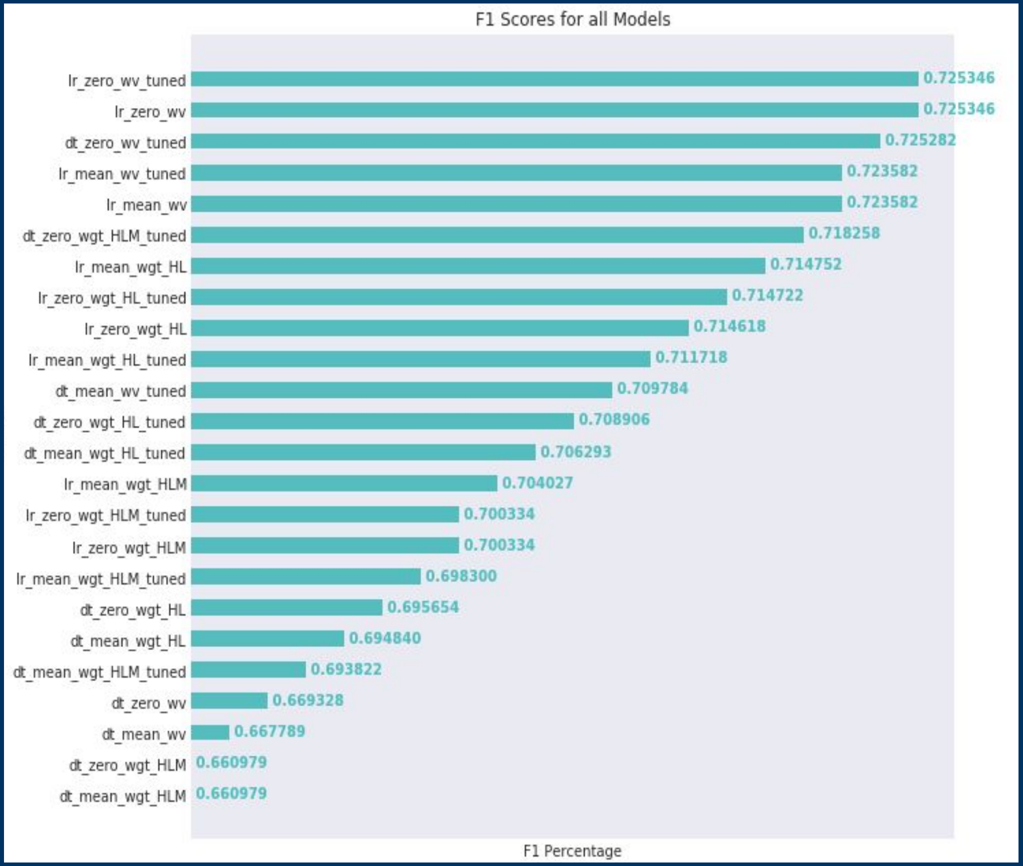
Algorithm Modeling Matrix

3 Transformation Types	2 Imputing Methods	2 Algorithms	2 Model Runs
Weighted Value	Nulls => Mean	Logistic Regression	Default
Hi Low	Nulls => 0	Decision Tree	Hypertuned
Hi Mid Low Missing			

Berkeley

Model Performance Review

- Transformations:
  - Weighted value > Bucketing
- Imputing method:
  - No discernible difference between zero and mean
- Algorithms:
  - Logistic Regression > Decision Tree overall
- Model Runs:
  - Tuning did not improve F1 very much

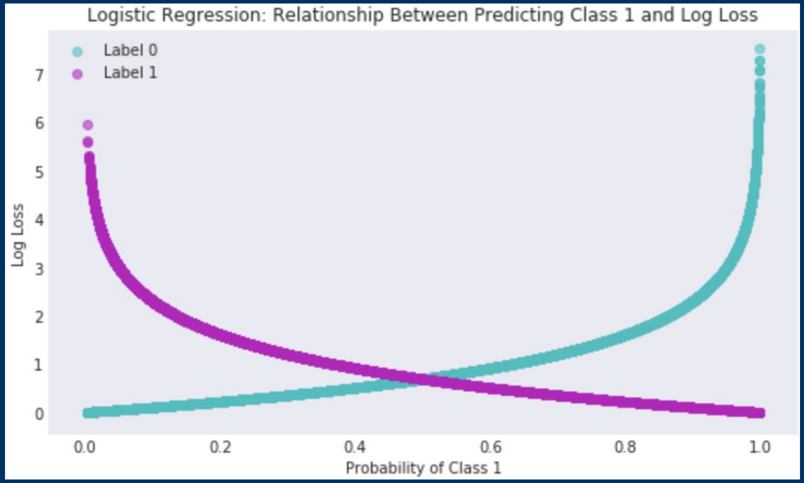


Berkeley

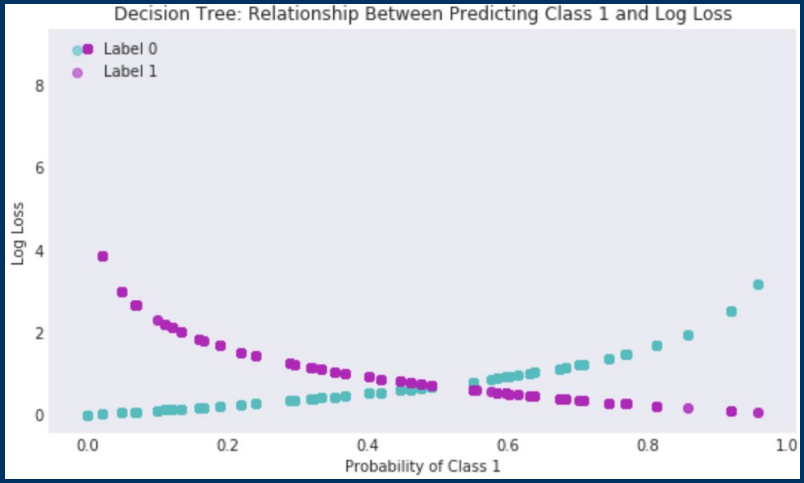
Log Loss equation for a single observation: $-y_i \cdot log(\hat{y}) - (1 - y_i) \cdot log(1 - \hat{y})$

- Increases exponentially based on how far the prediction is away from the actual value
- Penalizes predictions more if the prediction probability is further away from the actual value

Logistic Regression Log Loss = 0.562



Decision Tree Log Loss = 0.601



Berkeley

- **Overfitting –** wanted to make sure that we did not learn our model too well so employed bucketing techniques of the categorical data

- **Performance at Scale –** were able to deploy 8 of our models on the cloud with all transformations except for numeric log transformation

Categorical Transformation	Numeric Null Handling	Scaling Worked - LR: vs. DT
Weighted	impute 0	LR: No; DT: No
	inpute mean	LR: No; DT: No
Hi, Lo, Medium (binning) based on weights Nulls converted to M	impute 0	LR: Yes; DT: Yes
	inpute mean	LR: No; DT: No
Hi Lo (One hot encoding) 2 Nulls converted to L	impute 0	LR: Yes; DT: Yes
	inpute mean	LR: No; DT: No

Berkeley

- **Lazy Evaluation -** allows a process to wait to evaluate an expression (a "transformations") until the result is needed (by an "action")
  - Organize our steps for efficiency, and to let Spark handle the overhead of deciding in which order transformations happen

- **One-Hot Encoding, Binning and Dimensionality Reduction**
  - Map values with occurrence rates below a threshold
  - Update our binning rule to take into account confidence intervals based on the ratio of success
  - Refine whether or how to continue to bin multiple values into our "high, middle, low, etc." categories or one-hot encode with the occurrence rate or confidence thresholds only

Berkeley

- **Which machine learning algorithm produce the best predictions of CTR?**
    - Best Logistic Regression: Weighted value, zero imputation, parameter tuned (F1: 0.725)
    - Best Decision Tree: Weighted value, zero imputation, parameter tuned (F1: 0.725)
- **What metric(s) should we consider when we are defining "best" predictions of CTR?**
    - F1 score since it takes both false positives and false negatives into account without needing to weight them equally
- **How can we implement scalable machine learning algorithms that efficiently handle a large amount of data?**
    - We had partial success in that we scaled eight of the 24 models including all transformations (and parameter hypertuning as well)
    - Our best models with 1M rows need null value debugging to successfully scale

Berkeley

Thank you