W261 Final Project

Team 10

Youzhi (Chloe) Wu, Chun-Jen (Curtis) Lin, Eddie Zhu, Kai Qi Lim

Outline

- Introduction & Objective
- EDA
- Algorithm explanation and implementation
- Feature engineering
- Running the full dataset
- Application of course concepts

Marketing costs money...

- **Our objective is:**
  - To analyze the features of an ad
  - To predict if an ad would be clicked
  - To help increase ad click through rate (CTR)
  - To eventually, establish a targeted and effective marketing strategy

EDA

First look at raw data

0	1	2	3	4
0	3	5	8	32
0	0	0	null	null
0	null	-1	null	null
0	24	331	32	75
1	1	10	47	12
0	3	10	5	12
0	0	1	null	null
0	0	3	11	2
0	null	25	3	7
0	null	1	3	9
0	null	4	16	6
0	13	0	6	9
0	0	19	19	4
0	null	-1	null	null
0	null	28	12	17
0	5	0	1	3
0	null	3	null	null
0	null	307	1	1
0	null	485	65	7
0	null	-1	null	null

…

12	13	14	15	16
null	44	05db9164	38a947a1	42a04a35
null	null	05db9164	09e68b86	1b2c0594
null	null	8cf07265	3f0d3f28	d7ee147d
0	207	f473b8dc	942f9a8d	435c3859
null	2	05db9164	38d50e09	15e1524a
0	12	05db9164	90081f33	f30661f3
null	null	05db9164	dc1def19	5d1ca1e5
null	2	5a9ed9b0	78ccd99e	4d485115
0	7	05db9164	09e68b86	2794b521
1	9	68fd1e64	207b2d81	d73903c4
null	6	05db9164	a0e12995	b3d2f3df
0	1	05db9164	95e2d337	20e4194f
0	4	68fd1e64	3ab4d7f5	a4b38ecc
0	null	05db9164	a796837e	5c05f1ab
0	38	5a9ed9b0	08d6d899	9143c832
null	0	be589b51	207b2d81	8f113de9
null	null	8cf07265	38a947a1	4470baf4
null	1	05db9164	8db5bc37	4e1476b1
null	7	05db9164	1cfdf714	ea96c30a
null	null	05db9164	b961056b	86ed5799

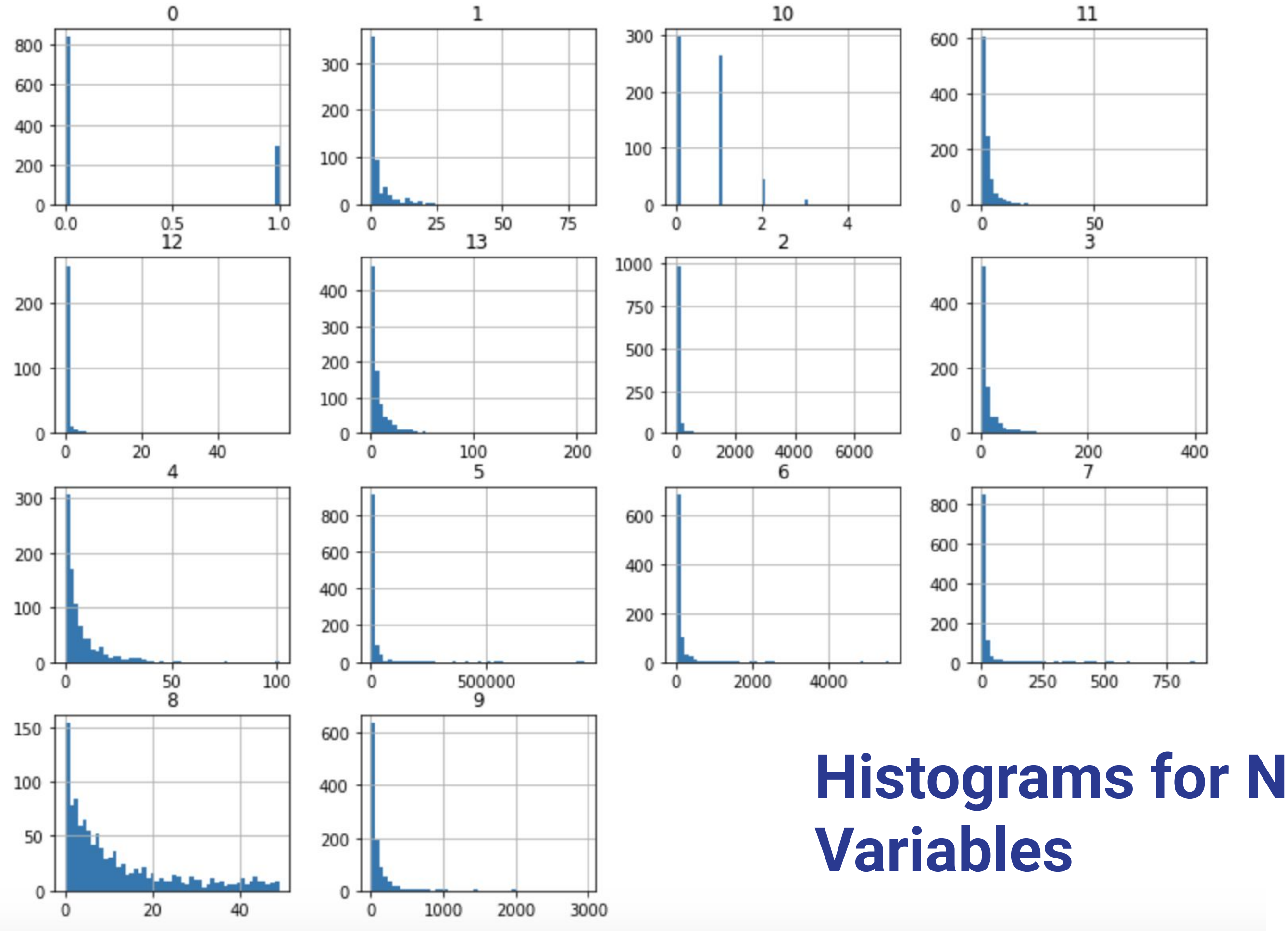
…

35	36	37	38	39
ad3062eb	423fab69	b258af68	null	null
ad3062eb	32c7478e	3900a598	e8b83407	396cda21
null	3a171ecb	e5fca70a	null	null
null	32c7478e	9fc6df89	9d93af03	91002aff
ad3062eb	32c7478e	df487a73	001f3601	c27f155b
null	32c7478e	d5c410bd	null	null
null	c3dc6cef	6bc2bf95	null	null
null	3a171ecb	bb90fac0	c243e98b	c12a6ac4
null	32c7478e	df707311	e8b83407	006de9be
null	32c7478e	cc92fe5a	001f3601	bc84f389
ad3062eb	3a171ecb	8548f4a1	9b3e8820	e75c9ae9
null	423fab69	16bb3de8	2bf691b1	d3b2f8c3
null	3a171ecb	65e74c52	c9f3bea7	f3ea27fd
null	32c7478e	8fc66e78	null	null
null	32c7478e	c0d61a5c	null	null
null	32c7478e	e4cf11c4	001f3601	032198c9
null	32c7478e	8d365d3b	null	null
null	be7c41b4	1aa84899	null	null
ad3062eb	423fab69	7939255b	cb079c2d	bc33111f
null	32c7478e	b0835adb	null	null

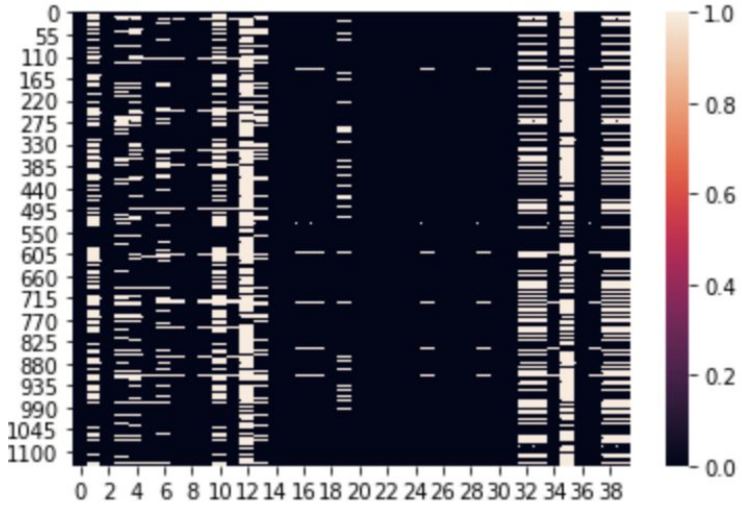
Target

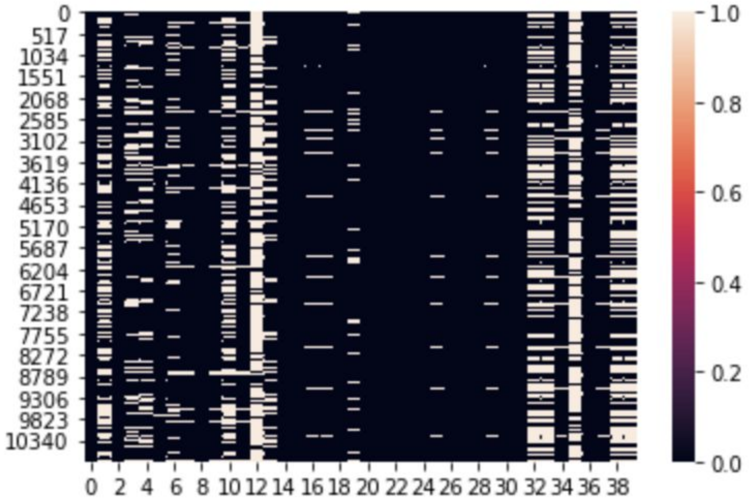
13 numeric variables

26 categorical variables



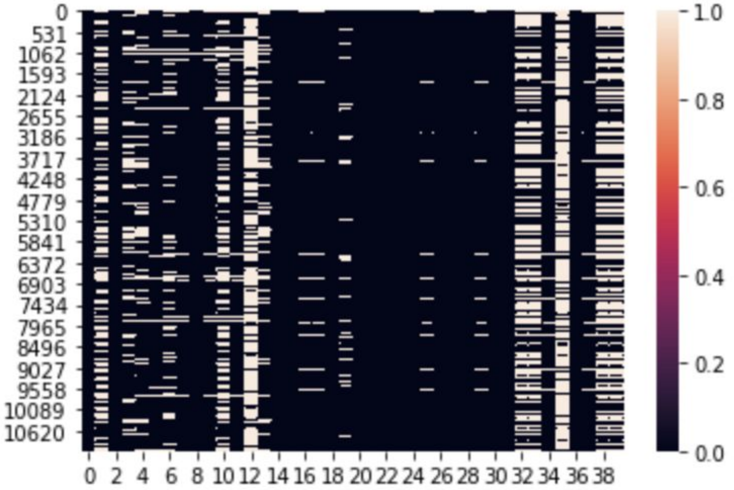
Histograms for Numeric Variables



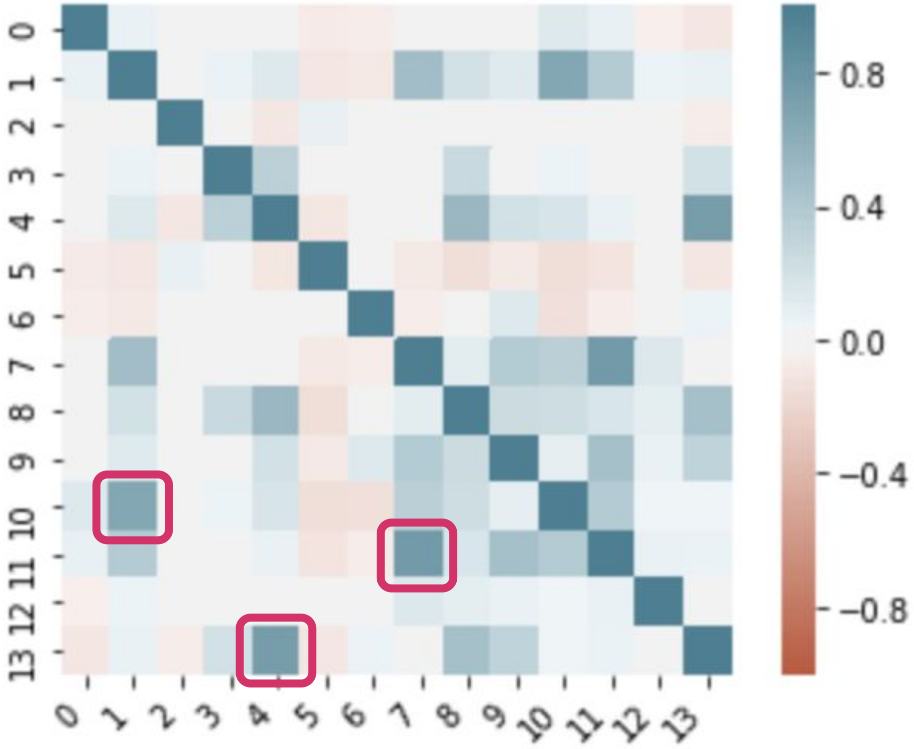


Heatmaps

- To compare similarity



Correlation Heatmap

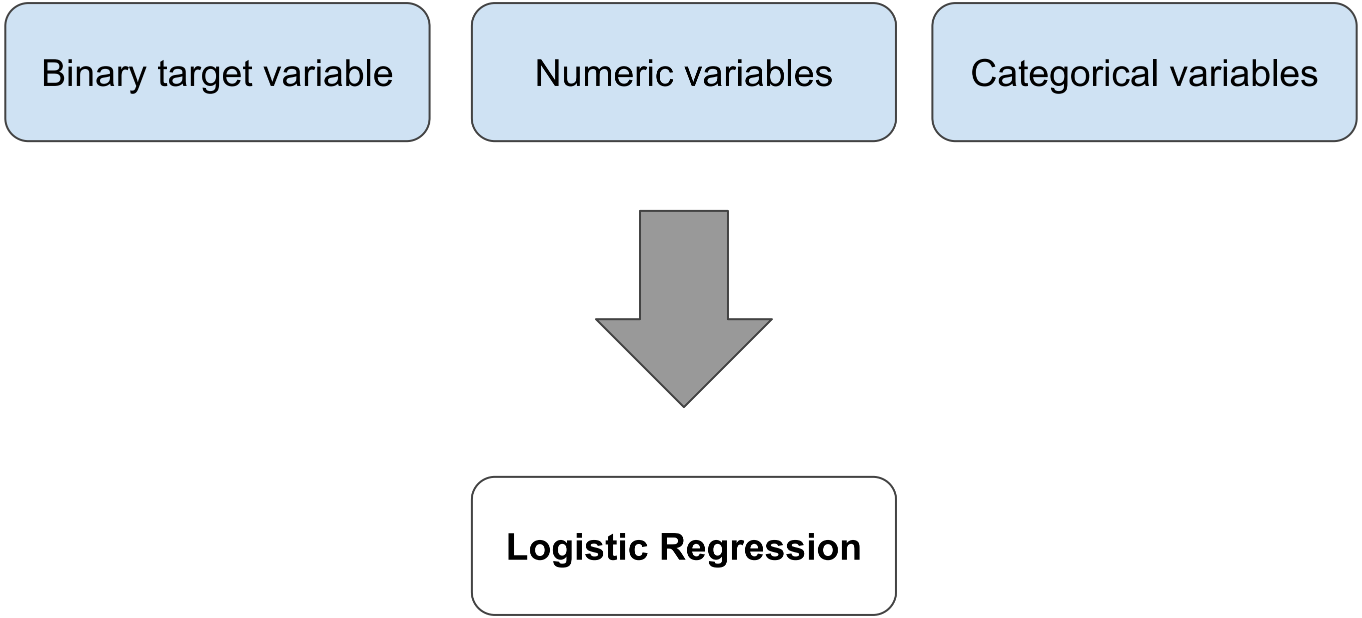


Strong positive correlation:

- Column 1 and 10
- Column 7 and 11
- Column 4 and 13

Algorithm Explanation

Model selection



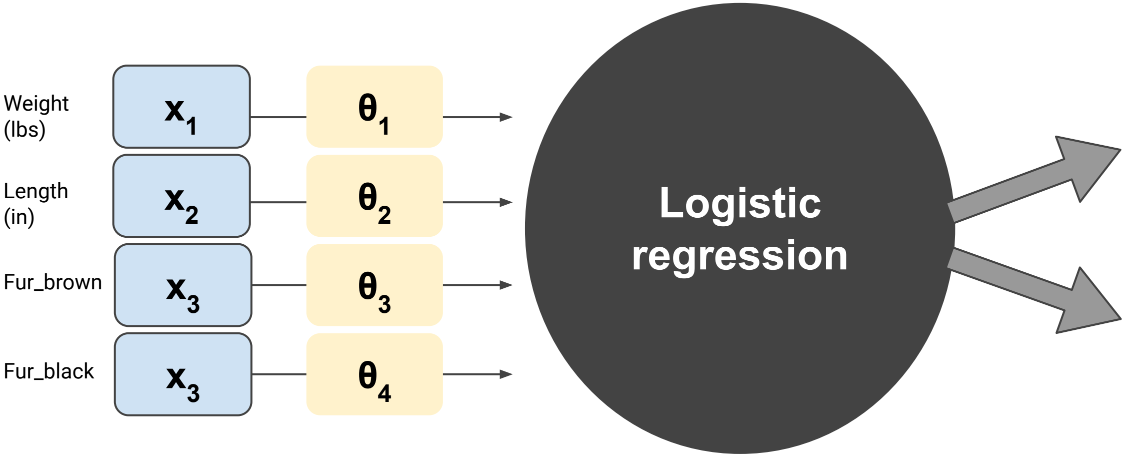
Mini toy dataset

Target	Weight (lbs)	Length (in)	Colour of fur
1	7.9	18.1	Brown
0	44.1	20	Black
0	55.1	27.5	Brown
1	8.2	17.5	Brown
1	9.0	18.5	Black

Binary

Numeric

Categorical



Logistic Regression

Probability that the animal is cat:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\theta \cdot \mathbf{x}_i'^T)}}$$

Probability that the animal is dog:

$$P(y = 0|x) = 1 - \frac{1}{1 + e^{-(\theta \cdot \mathbf{x}_i'^T)}}$$

Log loss as cost function:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y^i \cdot log(h_\theta(x_i)) + (1 - y^i) \cdot log(1 - h_\theta(x_i))]$$

$$h_\theta(x_i) = \frac{1}{1 + e^{-(\theta \cdot \mathbf{x}_i'^T)}}$$

$$\theta := \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

Gradient Descent:

$$g(\theta_j) = \frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^{n}(h_\theta(x_j^i) - y^i)x_j^i$$

$$\theta_{j+1} = \theta_j - \alpha \sum_{i=1}^{n}(h_\theta(x_j^i) - y^i)x_j^i$$

Implementation

Target	Weight (lbs)	Length (in)	Colour of fur
1	7.9	18.1	Brown
0	44.1	20	Black
0	55.1	27.5	Brown
1	8.2	17.5	Brown
1	9.0	18.5	Black

parse

onehot

```
[(array([2.06686276, 2.89591194, 0.        , 1.        ]), 1),
 (array([3.78645978, 2.99573227, 1.        , 0.        ]), 0),
 (array([4.00914972, 3.314186  , 0.        , 1.        ]), 0),
 (array([2.10413415, 2.86220088, 0.        , 1.        ]), 1),
 (array([2.19722458, 2.91777073, 1.        , 0.        ]), 1)]
```

normalize

```
[(array([-0.87684153, -0.61555096, -0.81649658,  0.81649658]), 1),
 (array([ 1.09183234, -0.00868224,  1.22474487, -1.22474487]), 0),
 (array([ 1.34677803,  1.92739226, -0.81649658,  0.81649658]), 0),
 (array([-0.83417152, -0.82050104, -0.81649658,  0.81649658]), 1),
 (array([-0.72759732, -0.48265802,  1.22474487, -1.22474487]), 1)]
```

```
BASELINE = np.array([0, 0, 0, 0, 0])
```

AugmentRDD

```
[((0.5,
   array([ 1.        , -0.87684153, -0.61555096, -0.81649658,  0.81649658])),
  1),
 ((0.5,
   array([ 1.        ,  1.09183234, -0.00868224,  1.22474487, -1.22474487])),
  0),
```

LGRDFit

AugmentRDD

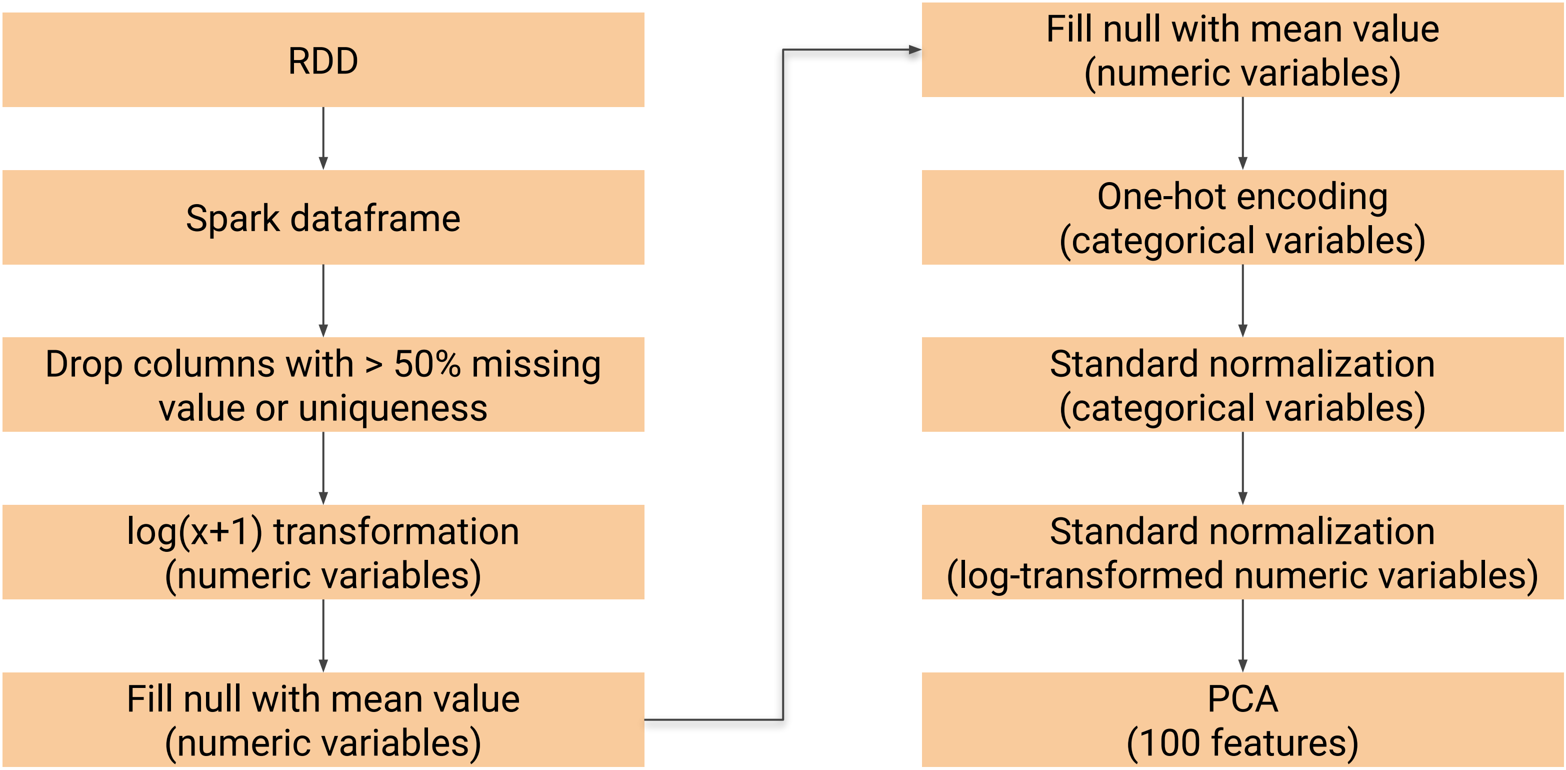
GDUpdate

LRLoss

```
STEP: 19
training loss: 0.6870787403218255
Model: [0.019, -0.089, -0.07, -0.015, 0.015]
----------
STEP: 20
training loss: 0.6869000224161442
Model: [0.02, -0.094, -0.073, -0.016, 0.016]
```

Feature Engineering

Flow of feature engineering



RDD to Spark dataframe

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	3	5	8	32	766	93	25	47	326	1	4	null	44	05db9164	38a947a1	42a04a35	7031bb66	25c83c98	7e0ccccf	15f8eb35
0	0	0	null	null	1571	5	7	5	13	0	3	null	null	05db9164	09e68b86	1b2c0594	d8d9a31d	25c83c98	7e0ccccf	9a4f2943
0	null	-1	null	null	18086	86	1	1	45	null	1	null	null	8cf07265	3f0d3f28	d7ee147d	e4f81cba	384874ce	fe6b92e5	928efe34
0	24	331	32	75	560	253	177	22	1965	1	22	0	207	f473b8dc	942f9a8d	435c3859	be42d44a	25c83c98	7e0ccccf	3f4ec687
1	1	10	47	12	2	2	1	15	60	1	1	null	2	05db9164	38d50e09	15e1524a	f438658e	f281d2a7	fbad5c96	9963c37d
0	3	10	5	12	568	15	3	11	12	1	1	0	12	05db9164	90081f33	f30661f3	d285863e	384874ce	fe6b92e5	e0218d0b
0	0	1	null	null	16428	null	null	5	null	0	null	null	null	05db9164	dc1def19	5d1ca1e5	196060ff	4cf72387	fe6b92e5	c0251c88
0	0	3	11	2	2059	63	7	39	63	0	1	null	2	5a9ed9b0	78ccd99e	4d485115	dd51f47e	384874ce	3bf701e7	6990f900
0	null	25	3	7	15719	44	25	7	42	null	1	0	7	05db9164	09e68b86	2794b521	d62b7a28	25c83c98	7e0ccccf	24c48926
0	null	1	3	9	2727	13	19	11	67	null	2	1	9	68fd1e64	207b2d81	d73903c4	77a972c7	25c83c98	null	d2dbdfe6
0	null	4	16	6	10075	null	0	6	16	null	0	null	6	05db9164	a0e12995	b3d2f3df	811a5b41	25c83c98	fe6b92e5	ed70802a
0	13	0	6	9	0	1	197	18	191	1	12	0	1	05db9164	95e2d337	20e4194f	c5455c5f	25c83c98	7e0ccccf	ac2d4799
0	0	19	19	4	1496	31	1	18	18	0	1	0	4	68fd1e64	3ab4d7f5	a4b38ecc	27d8fdb5	25c83c98	null	87e1825e
0	null	-1	null	null	13865	133	5	11	100	null	2	0	null	05db9164	a796837e	5c05f1ab	97ce69e9	25c83c98	7e0ccccf	c9255aae
0	null	28	12	17	2836	279	3	15	249	null	2	0	38	5a9ed9b0	08d6d899	9143c832	f56b7dd5	25c83c98	fe6b92e5	410f6b58
0	5	0	1	3	4	1	5	25	65	2	2	null	0	be589b51	207b2d81	8f113de9	f6dcfdcf	4cf72387	null	d2dbdfe6
0	null	3	null	null	35526	null	0	4	2	null	0	null	null	8cf07265	38a947a1	4470baf4	8c8a4c47	25c83c98	3bf701e7	ce4f7f55
0	null	307	1	1	null	null	0	1	4	null	0	null	1	05db9164	8db5bc37	4e1476b1	2c20fcf8	4cf72387	fe6b92e5	d356c7e6
0	null	485	65	7	80	140	4	7	128	null	1	null	7	05db9164	1cfdf714	ea96c30a	a7678b46	384874ce	fbad5c96	90a2c015
0	null	-1	null	null	25489	null	null	8	null	null	null	null	null	05db9164	b961056b	86ed5799	e1281c2c	4cf72387	3bf701e7	ce4f7f55

Total 40 columns

Drop columns with > 50% null or uniqueness

0	2	3	4	5	6	7	8	9	11	13	14	15	18	19	21	22	27	30	36
0	5	8	32	766	93	25	47	326	4	44	05db9164	38a947a1	25c83c98	7e0ccccf	f504a6f4	a73ee510	b28479f6	e5ba7672	423fab69
0	0	null	null	1571	5	7	5	13	3	null	05db9164	09e68b86	25c83c98	7e0ccccf	5b392875	a73ee510	1adce6ef	e5ba7672	32c7478e
0	-1	null	null	18086	86	1	1	45	1	null	8cf07265	3f0d3f28	384874ce	fe6b92e5	5b392875	a73ee510	32813e21	e5ba7672	3a171ecb
0	331	32	75	560	253	177	22	1965	22	207	f473b8dc	942f9a8d	25c83c98	7e0ccccf	0b153874	a73ee510	1adce6ef	27c07bd6	32c7478e
1	10	47	12	2	2	1	15	60	1	2	05db9164	38d50e09	f281d2a7	fbad5c96	5b392875	a73ee510	b28479f6	d4bb7bd8	32c7478e
0	10	5	12	568	15	3	11	12	1	12	05db9164	90081f33	384874ce	fe6b92e5	0b153874	a73ee510	b28479f6	07c540c4	32c7478e
0	1	null	null	16428	null	null	5	null	null	null	05db9164	dc1def19	4cf72387	fe6b92e5	0b153874	a73ee510	ad1cc976	e5ba7672	c3dc6cef
0	3	11	2	2059	63	7	39	63	1	2	5a9ed9b0	78ccd99e	384874ce	3bf701e7	37e4aa92	a73ee510	f862f261	e5ba7672	3a171ecb
0	25	3	7	15719	44	25	7	42	1	7	05db9164	09e68b86	25c83c98	7e0ccccf	5b392875	a73ee510	b28479f6	e5ba7672	32c7478e
0	1	3	9	2727	13	19	11	67	2	9	68fd1e64	207b2d81	25c83c98	null	37e4aa92	a73ee510	07d13a8f	3486227d	32c7478e
0	4	16	6	10075	null	0	6	16	0	6	05db9164	a0e12995	25c83c98	fe6b92e5	5b392875	a73ee510	b28479f6	776ce399	3a171ecb
0	0	6	9	0	1	197	18	191	12	1	05db9164	95e2d337	25c83c98	7e0ccccf	56563555	a73ee510	07d13a8f	8efede7f	423fab69
0	19	19	4	1496	31	1	18	18	1	4	68fd1e64	3ab4d7f5	25c83c98	null	0b153874	a73ee510	dcd762ee	d4bb7bd8	3a171ecb
0	-1	null	null	13865	133	5	11	100	2	null	05db9164	a796837e	25c83c98	7e0ccccf	0b153874	a73ee510	07d13a8f	e5ba7672	32c7478e
0	28	12	17	2836	279	3	15	249	2	38	5a9ed9b0	08d6d899	25c83c98	fe6b92e5	0b153874	a73ee510	b28479f6	3486227d	32c7478e
0	0	1	3	4	1	5	25	65	2	0	be589b51	207b2d81	4cf72387	null	f504a6f4	a73ee510	64c94865	3486227d	32c7478e
0	3	null	null	35526	null	0	4	2	0	null	8cf07265	38a947a1	25c83c98	3bf701e7	0b153874	a73ee510	b28479f6	1e88c74f	32c7478e
0	307	1	1	null	null	0	1	4	0	1	05db9164	8db5bc37	4cf72387	fe6b92e5	f504a6f4	7cc72ec2	64c94865	2005abd1	be7c41b4
0	485	65	7	80	140	4	7	128	1	7	05db9164	1cfdf714	384874ce	fbad5c96	0b153874	a73ee510	1adce6ef	e5ba7672	423fab69
0	-1	null	null	25489	null	null	8	null	null	null	05db9164	b961056b	4cf72387	3bf701e7	0b153874	a73ee510	b28479f6	07c540c4	32c7478e

Columns with > 50% null value: '1', '10', '12', '32', '33', '35', '38', '39'

Columns with > 50% uniqueness: '16', '17', '20', '23', '24', '25', '26', '28', '29', '31', '34', '37'

Log(x+1) transformation (numeric)

2	3	4	5
5.0	8.0	32.0	766.0
0.0	null	null	1571.0
-1.0	null	null	18086.0
331.0	32.0	75.0	560.0
10.0	47.0	12.0	2.0
10.0	5.0	12.0	568.0
1.0	null	null	16428.0
3.0	11.0	2.0	2059.0
25.0	3.0	7.0	15719.0
1.0	3.0	9.0	2727.0
4.0	16.0	6.0	10075.0
0.0	6.0	9.0	0.0
19.0	19.0	4.0	1496.0
-1.0	null	null	13865.0
28.0	12.0	17.0	2836.0
0.0	1.0	3.0	4.0
3.0	null	null	35526.0
307.0	1.0	1.0	null
485.0	65.0	7.0	80.0
-1.0	null	null	25489.0

log_2	log_3	log_4	log_5
1.791759469228055	2.1972245773362196	3.4965075614664802	6.642486801367256
0.0	null	null	7.360103972989152
null	null	null	9.802948727157515
5.805134968916488	3.4965075614664802	4.330733340286331	6.329720905522696
2.3978952727983707	3.871201010907891	2.5649493574615367	1.0986122886681098
2.3978952727983707	1.791759469228055	2.5649493574615367	6.343880434126331
0.6931471805599453	null	null	9.706803344906337
1.3862943611198906	2.4849066497880004	1.0986122886681098	7.630461261783627
3.258096538021482	1.3862943611198906	2.0794415416798357	9.662689065983198
0.6931471805599453	1.3862943611198906	2.302585092994046	7.911324018963353
1.6094379124341003	2.833213344056216	1.9459101490553132	9.2179116374725
0.0	1.9459101490553132	2.302585092994046	0.0
2.995732273553991	2.995732273553991	1.6094379124341003	7.311218384419629
null	null	null	9.537195079502435
3.367295829986474	2.5649493574615367	2.8903717578961645	7.950502434808851
0.0	0.6931471805599453	1.3862943611198906	1.6094379124341003
1.3862943611198906	null	null	10.478048249762029
5.730099782973574	0.6931471805599453	0.6931471805599453	null
6.186208623900494	4.189654742026425	2.0794415416798357	4.394449154672439
null	null	null	10.146041497370161

Fill null with mean (numeric)

log_2	log_3	log_4
1.791759469228055	2.1972245773362196	3.4965075614664802
0.0	null	null
null	null	null
5.8051349689164888	3.4965075614664802	4.330733340286331
2.3978952727983707	3.871201010907891	2.5649493574615367
2.3978952727983707	1.791759469228055	2.5649493574615367
0.6931471805599453	null	null
1.3862943611198906	2.4849066497880004	1.0986122886681098
3.258096538021482	1.3862943611198906	2.0794415416798357
0.6931471805599453	1.3862943611198906	2.302585092994046
1.6094379124341003	2.833213344056216	1.9459101490553132
0.0	1.9459101490553132	2.302585092994046
2.995732273553991	2.995732273553991	1.6094379124341003
null	null	null
3.367295829986474	2.5649493574615367	2.890371757896165
0.0	0.6931471805599453	1.3862943611198906
1.3862943611198906	null	null
5.730099782973574	0.6931471805599453	0.6931471805599453
6.186208623900494	4.189654742026425	2.0794415416798357
null	null	null

log_2_imputed	log_3_imputed	log_4_imputed
1.791759469228055	2.1972245773362196	3.4965075614664802
0.0	2.1407438041148916	1.7041792570817396
2.3878927248048827	2.1407438041148916	1.7041792570817396
5.8051349689164888	3.4965075614664802	4.330733340286331
2.3978952727983707	3.871201010907891	2.5649493574615367
2.3978952727983707	1.791759469228055	2.5649493574615367
0.6931471805599453	2.1407438041148916	1.7041792570817396
1.3862943611198906	2.4849066497880004	1.0986122886681098
3.258096538021482	1.3862943611198906	2.0794415416798357
0.6931471805599453	1.3862943611198906	2.302585092994046
1.6094379124341003	2.833213344056216	1.9459101490553132
0.0	1.9459101490553132	2.302585092994046
2.995732273553991	2.995732273553991	1.6094379124341003
2.3878927248048827	2.1407438041148916	1.7041792570817396
3.367295829986474	2.5649493574615367	2.890371757896165
0.0	0.6931471805599453	1.3862943611198906
1.3862943611198906	2.1407438041148916	1.7041792570817396
5.730099782973574	0.6931471805599453	0.6931471805599453
6.186208623900494	4.189654742026425	2.0794415416798357
2.3878927248048827	2.1407438041148916	1.7041792570817396

Fill null with zero for (categorical)

14	15	18	19	21
05db9164	38a947a1	25c83c98	7e0ccccf	f504a6f4
05db9164	09e68b86	25c83c98	7e0ccccf	5b392875
8cf07265	3f0d3f28	384874ce	fe6b92e5	5b392875
f473b8dc	942f9a8d	25c83c98	7e0ccccf	0b153874
05db9164	38d50e09	f281d2a7	fbad5c96	5b392875
05db9164	90081f33	384874ce	fe6b92e5	0b153874
05db9164	dc1def19	4cf72387	fe6b92e5	0b153874
5a9ed9b0	78ccd99e	384874ce	3bf701e7	37e4aa92
05db9164	09e68b86	25c83c98	7e0ccccf	5b392875
68fd1e64	207b2d81	25c83c98	null	37e4aa92
05db9164	a0e12995	25c83c98	fe6b92e5	5b392875
05db9164	95e2d337	25c83c98	7e0ccccf	56563555
68fd1e64	3ab4d7f5	25c83c98	null	0b153874
05db9164	a796837e	25c83c98	7e0ccccf	0b153874
5a9ed9b0	08d6d899	25c83c98	fe6b92e5	0b153874
be589b51	207b2d81	4cf72387	null	f504a6f4
8cf07265	38a947a1	25c83c98	3bf701e7	0b153874
05db9164	8db5bc37	4cf72387	fe6b92e5	f504a6f4
05db9164	1cfdf714	384874ce	fbad5c96	0b153874
05db9164	b961056b	4cf72387	3bf701e7	0b153874

14	15	18	19	21
05db9164	38a947a1	25c83c98	7e0ccccf	f504a6f4
05db9164	09e68b86	25c83c98	7e0ccccf	5b392875
8cf07265	3f0d3f28	384874ce	fe6b92e5	5b392875
f473b8dc	942f9a8d	25c83c98	7e0ccccf	0b153874
05db9164	38d50e09	f281d2a7	fbad5c96	5b392875
05db9164	90081f33	384874ce	fe6b92e5	0b153874
05db9164	dc1def19	4cf72387	fe6b92e5	0b153874
5a9ed9b0	78ccd99e	384874ce	3bf701e7	37e4aa92
05db9164	09e68b86	25c83c98	7e0ccccf	5b392875
68fd1e64	207b2d81	25c83c98	0	37e4aa92
05db9164	a0e12995	25c83c98	fe6b92e5	5b392875
05db9164	95e2d337	25c83c98	7e0ccccf	56563555
68fd1e64	3ab4d7f5	25c83c98	0	0b153874
05db9164	a796837e	25c83c98	7e0ccccf	0b153874
5a9ed9b0	08d6d899	25c83c98	fe6b92e5	0b153874
be589b51	207b2d81	4cf72387	0	f504a6f4
8cf07265	38a947a1	25c83c98	3bf701e7	0b153874
05db9164	8db5bc37	4cf72387	fe6b92e5	f504a6f4
05db9164	1cfdf714	384874ce	fbad5c96	0b153874
05db9164	b961056b	4cf72387	3bf701e7	0b153874

One-hot encoding (categorical)

15	18	19
38a947a1	25c83c98	7e0ccccf
09e68b86	25c83c98	7e0ccccf
3f0d3f28	384874ce	fe6b92e5
942f9a8d	25c83c98	7e0ccccf
38d50e09	f281d2a7	fbad5c96
90081f33	384874ce	fe6b92e5
dc1def19	4cf72387	fe6b92e5
78ccd99e	384874ce	3bf701e7
09e68b86	25c83c98	7e0ccccf
207b2d81	25c83c98	0
a0e12995	25c83c98	fe6b92e5
95e2d337	25c83c98	7e0ccccf
3ab4d7f5	25c83c98	0
a796837e	25c83c98	7e0ccccf
08d6d899	25c83c98	fe6b92e5
207b2d81	4cf72387	0
38a947a1	25c83c98	3bf701e7
8db5bc37	4cf72387	fe6b92e5
1cfdf714	384874ce	fbad5c96
b961056b	4cf72387	3bf701e7

15_indexed	18_indexed	19_indexed
0.0	0.0	0.0
5.0	0.0	0.0
40.0	3.0	2.0
30.0	0.0	0.0
3.0	7.0	1.0
21.0	3.0	2.0
159.0	1.0	2.0
22.0	3.0	5.0
5.0	0.0	0.0
1.0	0.0	3.0
20.0	0.0	2.0
18.0	0.0	0.0
66.0	0.0	3.0
15.0	0.0	0.0
11.0	0.0	2.0
1.0	1.0	3.0
0.0	0.0	5.0
27.0	1.0	2.0
4.0	3.0	1.0
42.0	1.0	5.0

15_indexed_encoded	18_indexed_encoded	19_indexed_encoded
(210,[0],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[5],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[40],[1.0])	(26,[3],[1.0])	(7,[2],[1.0])
(210,[30],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[3],[1.0])	(26,[7],[1.0])	(7,[1],[1.0])
(210,[21],[1.0])	(26,[3],[1.0])	(7,[2],[1.0])
(210,[159],[1.0])	(26,[1],[1.0])	(7,[2],[1.0])
(210,[22],[1.0])	(26,[3],[1.0])	(7,[5],[1.0])
(210,[5],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[1],[1.0])	(26,[0],[1.0])	(7,[3],[1.0])
(210,[20],[1.0])	(26,[0],[1.0])	(7,[2],[1.0])
(210,[18],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[66],[1.0])	(26,[0],[1.0])	(7,[3],[1.0])
(210,[15],[1.0])	(26,[0],[1.0])	(7,[0],[1.0])
(210,[11],[1.0])	(26,[0],[1.0])	(7,[2],[1.0])
(210,[1],[1.0])	(26,[1],[1.0])	(7,[3],[1.0])
(210,[0],[1.0])	(26,[0],[1.0])	(7,[5],[1.0])
(210,[27],[1.0])	(26,[1],[1.0])	(7,[2],[1.0])
(210,[4],[1.0])	(26,[3],[1.0])	(7,[1],[1.0])
(210,[42],[1.0])	(26,[1],[1.0])	(7,[5],[1.0])

Standard normalization (categorical)

```
+--------------------+--------------------+
|         CatFeatures|  scaled_categorical|
+--------------------+--------------------+
|(313,[0,210,236,2...|[2.81742580305247...|
|(313,[5,210,236,2...|[-0.3546214836349...|
|(313,[40,213,238,...|[-0.3546214836349...|
|(313,[30,210,236,...|[-0.3546214836349...|
|(313,[3,217,237,2...|[-0.3546214836349...|
|(313,[21,213,238,...|[-0.3546214836349...|
|(313,[159,211,238...|[-0.3546214836349...|
|(313,[22,213,241,...|[-0.3546214836349...|
|(313,[5,210,236,2...|[-0.3546214836349...|
|(313,[1,210,239,2...|[-0.3546214836349...|
|(313,[20,210,238,...|[-0.3546214836349...|
|(313,[18,210,236,...|[-0.3546214836349...|
|(313,[66,210,239,...|[-0.3546214836349...|
|(313,[15,210,236,...|[-0.3546214836349...|
|(313,[11,210,238,...|[-0.3546214836349...|
|(313,[1,211,239,2...|[-0.3546214836349...|
|(313,[0,210,241,2...|[2.81742580305247...|
|(313,[27,211,238,...|[-0.3546214836349...|
|(313,[4,213,237,2...|[-0.3546214836349...|
|(313,[42,211,241,...|[-0.3546214836349...|
+--------------------+--------------------+
```

Standard normalization (numeric)

NumFeatures	normed_log_numeric
[1.79175946922805...	[0.14110795622397...
[0.0,2.1407438041...	[0.0,0.2332649368...
[2.38789272480488...	[0.19610839202661...
[5.80513496891648...	[0.35465531617691...
[2.39789527279837...	[0.31905825897196...
[2.39789527279837...	[0.25977321946075...
[0.69314718055994...	[0.05942545628612...
[1.38629436111989...	[0.12564346303016...
[3.25809653802148...	[0.25870153799060...
[0.69314718055994...	[0.06415826869052...
[1.60943791243410...	[0.14324580018990...
[0.0,1.9459101490...	[0.0,0.2163390314...
[2.99573227355399...	[0.28975058317799...
[2.38789272480488...	[0.18705662294283...
[3.36729582998647...	[0.25385015210811...
[0.0,0.6931471805...	[0.0,0.1123564859...
[1.38629436111989...	[0.11835647694140...
[5.73009978297357...	[0.57232154095446...
[6.18620862390049...	[0.52417113302866...
[2.38789272480488...	[0.19393173881430...

Principal component analysis (PCA)

features	target	pca_features
[0.14110795622397...	0.0	[0.24030868333957...
[0.0,0.2332649368...	0.0	[0.55408339984934...
[0.19610839202661...	0.0	[-0.0771215164390...
[0.35465531617691...	0.0	[0.33049506140704...
[0.31905825897196...	1.0	[0.50464937609451...
[0.25977321946075...	0.0	[0.14319884895544...
[0.05942545628612...	0.0	[-26.694044964925...
[0.12564346303016...	0.0	[-0.5852433611828...
[0.25870153799060...	0.0	[0.61161878947055...
[0.06415826869052...	0.0	[0.30650971368275...
[0.14324580018990...	0.0	[0.04388051899661...
[0.0,0.2163390314...	0.0	[0.35636523289388...
[0.28975058317799...	0.0	[-0.4144039944670...
[0.18705662294283...	0.0	[0.24722259338614...
[0.25385015210811...	0.0	[-0.0406424696712...
[0.0,0.1123564859...	0.0	[0.59931500628918...
[0.11835647694140...	0.0	[-0.6609948525699...
[0.57232154095446...	0.0	[-0.5343917345447...
[0.52417113302866...	0.0	[0.03162110778649...
[0.19393173881430...	0.0	[-0.3780864784995...

Final dataframe for logistic regression

pca_features	target
[0.24030868333957...	0.0
[0.55408339984934...	0.0
[-0.0771215164390...	0.0
[0.33049506140704...	0.0
[0.50464937609451...	1.0
[0.14319884895544...	0.0
[-26.694044964925...	0.0
[-0.5852433611828...	0.0
[0.61161878947055...	0.0
[0.30650971368275...	0.0
[0.04388051899661...	0.0
[0.35636523289388...	0.0
[-0.4144039944670...	0.0
[0.24722259338614...	0.0
[-0.0406424696712...	0.0
[0.59931500628918...	0.0
[-0.6609948525699...	0.0
[-0.5343917345447...	0.0
[0.03162110778649...	0.0
[-0.3780864784995...	0.0

Logistic regression in small and middle scale

Statistical methods of model evaluation

Prediction Accuracy

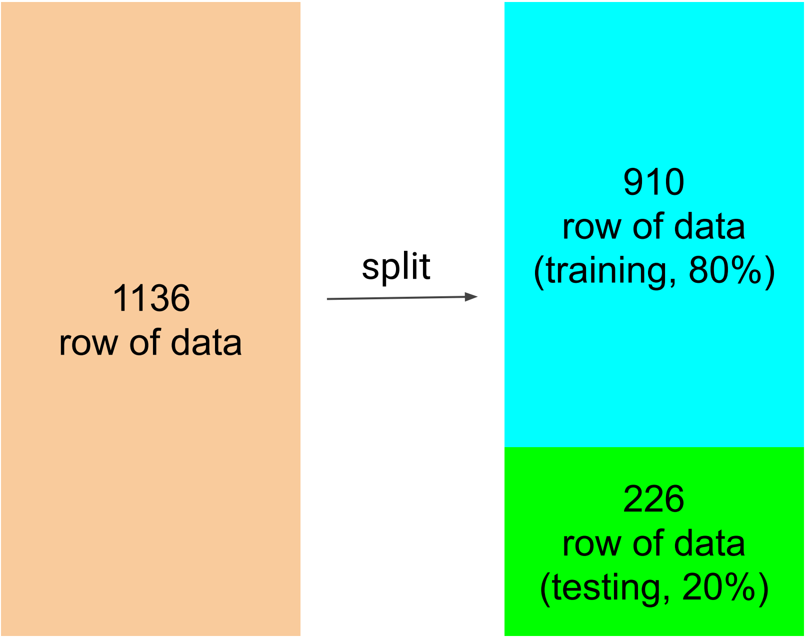
$$\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

$TP: true\ positive, TN: true\ negative, FP: false\ positive, FN: false\ negative$

Log loss

$$\text{Log}\ loss: J(\theta) = -\frac{1}{n}\sum_{k=0}^{n}[y_i \cdot \log{(h_\theta(x_i))} + (1 - y_i\ ) \cdot \log{(1 - h_\theta(x_i))}]$$

Small scale (~1000 row of data)



Prediction accuracy: 0.743

Log loss: 0.739

Middle scale (~10000 row of data)

10849
row of data
(training)

11131
row of data
(testing)

Prediction accuracy: 0.727

Log loss: 0.619

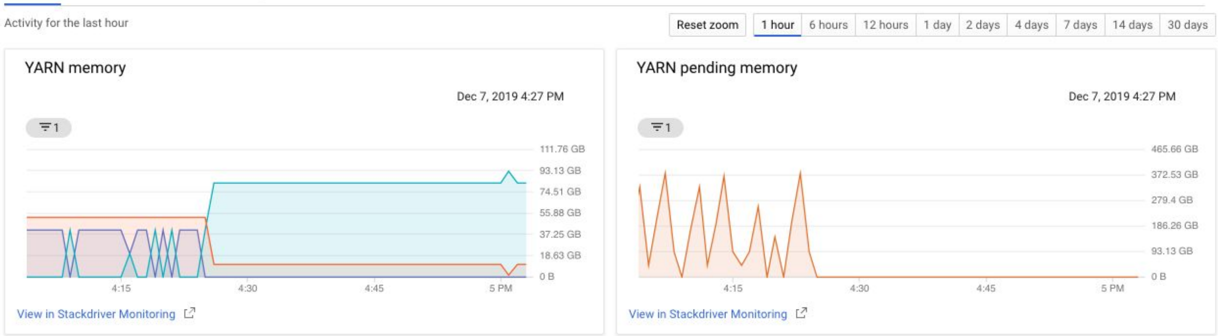
Running the full dataset & GCP

GCP is powerful but ...

- Spin up a cluster with enough memory
  We completely underestimated the amount of calculation involved in the full dataset. The consequence is all sorts of trials and errors.

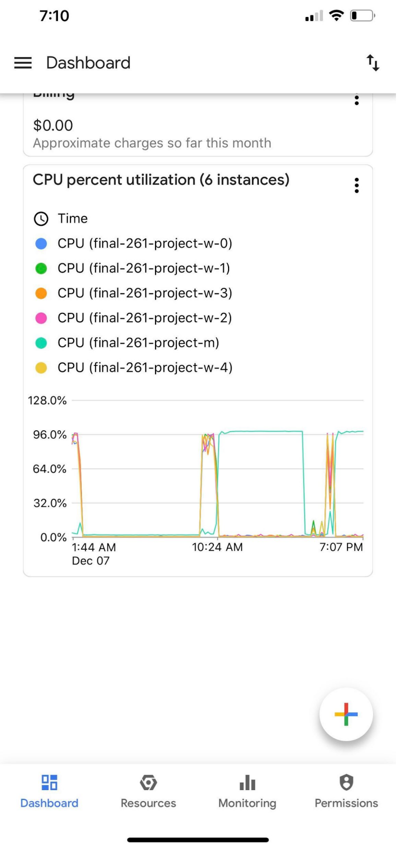
```
[*]: # perform one-hot encoding
     from pyspark.ml.feature import OneHotEncoderModel
     from pyspark.ml import Pipeline
     from pyspark.ml.feature import StringIndexer, OneHotEncoderEstimator, VectorAssembler

     train_hot = train_df_impute

     indexers = [StringIndexer(inputCol=col, outputCol="{0}_indexed".format(col)) for col in list(train_hot.columns[12:32])]

     encoder = OneHotEncoderEstimator(inputCols=[indexer.getOutputCol() for indexer in indexers],
                                      outputCols=["{0}_encoded".format(indexer.getOutputCol()) for indexer in indexers])
     cat_assembler = VectorAssembler(inputCols=encoder.getOutputCols(),outputCol="CatFeatures")

     pipeline = Pipeline(stages=indexers + [encoder, cat_assembler])
     train_onehot = pipeline.fit(train_hot).transform(train_hot)
     train_onehot.show()

     ---------------------------------------------------------------------------
     Py4JJavaError                             Traceback (most recent call last)
     /usr/lib/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
          62         try:
     ---> 63             return f(*a, **kw)
          64         except py4j.protocol.Py4JJavaError as e:

     /usr/lib/spark/python/lib/py4j-0.10.7-src.zip/py4j/protocol.py in get_return_value(answer, gateway_client, target_id,
     name)
         327                     "An error occurred while calling {0}{1}{2}.\n".
     --> 328                     format(target_id, ".", name), value)
         329             else:

     Py4JJavaError: An error occurred while calling o218.fit.
     : java.lang.IllegalArgumentException: requirement failed: Cannot have an empty string for name.
             at scala.Predef$.require(Predef.scala:224)
             at org.apache.spark.ml.attribute.Attribute$$anonfun$5.apply(attributes.scala:33)
             at org.apache.spark.ml.attribute.Attribute$$anonfun$5.apply(attributes.scala:32)
             at scala.Option.foreach(Option.scala:257)
```

```
Py4JJavaError: An error occurred while calling o215.fit.
: java.lang.OutOfMemoryError: GC overhead limit exceeded
        at scala.collection.immutable.HashMap$HashTrieMap.updated0(HashMap.scala:324)
        at scala.collection.immutable.HashMap$HashTrieMap.updated0(HashMap.scala:322)
        at scala.collection.immutable.HashMap$HashTrieMap.updated0(HashMap.scala:322)
        at scala.collection.immutable.HashMap.$plus(HashMap.scala:60)
        at scala.collection.immutable.HashMap.$plus(HashMap.scala:37)
        at scala.collection.mutable.MapBuilder.$plus$eq(MapBuilder.scala:29)
        at scala.collection.mutable.MapBuilder.$plus$eq(MapBuilder.scala:25)
        at scala.collection.TraversableOnce$$anonfun$toMap$1.apply(TraversableOnce.scala:317)
        at scala.collection.TraversableOnce$$anonfun$toMap$1.apply(TraversableOnce.scala:316)
        at scala.collection.IndexedSeqOptimized$class.foreach(IndexedSeqOptimized.scala:33)
        at scala.collection.mutable.ArrayOps$ofRef.foreach(ArrayOps.scala:186)
        at scala.collection.TraversableOnce$class.toMap(TraversableOnce.scala:316)
        at scala.collection.mutable.ArrayOps$ofRef.toMap(ArrayOps.scala:186)
        at org.apache.spark.rdd.PairRDDFunctions$$anonfun$countByKey$1.apply(PairRDDFunctions.sca
        at org.apache.spark.rdd.PairRDDFunctions$$anonfun$countByKey$1.apply(PairRDDFunctions.sca
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
        at org.apache.spark.rdd.RDD.withScope(RDD.scala:363)
        at org.apache.spark.rdd.PairRDDFunctions.countByKey(PairRDDFunctions.scala:369)
        at org.apache.spark.rdd.RDD$$anonfun$countByValue$1.apply(RDD.scala:1214)
        at org.apache.spark.rdd.RDD$$anonfun$countByValue$1.apply(RDD.scala:1214)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
        at org.apache.spark.rdd.RDD.withScope(RDD.scala:363)
        at org.apache.spark.rdd.RDD.countByValue(RDD.scala:1213)
        at org.apache.spark.ml.feature.StringIndexer.fit(StringIndexer.scala:140)
        at org.apache.spark.ml.feature.StringIndexer.fit(StringIndexer.scala:109)
        at sun.reflect.GeneratedMethodAccessor65.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
```



GCP works in mysterious ways ...

- It would have been nice to understand how GCP works
- How master assigns jobs to workers
- Which jobs are performed by master node
- Which jobs are performed by workers



GCP error message is a mercy ...

- So happy to see an error message
  - Many time, it stopped working without any message
  - Kernel dead without warning
  - Kept running without no result

```
    327                             "An error occurred while calling {0}{1}{2}.\n".
--> 328                             format(target_id, ".", name), value)
    329                     else:
    330                         raise Py4JError(

Py4JJavaError: An error occurred while calling o233.fit.
: java.lang.OutOfMemoryError: GC overhead limit exceeded
	at scala.collection.mutable.HashMap.createNewEntry(HashMap.scala:163)
	at scala.collection.mutable.HashMap.createNewEntry(HashMap.scala:40)
	at scala.collection.mutable.HashTable$class.findOrAddEntry(HashTable.scala:169)
	at scala.collection.mutable.HashMap.findOrAddEntry(HashMap.scala:40)
	at scala.collection.mutable.HashMap.put(HashMap.scala:107)
	at org.apache.spark.sql.types.MetadataBuilder.put(Metadata.scala:286)
	at org.apache.spark.sql.types.MetadataBuilder.putString(Metadata.scala:260)
	at org.apache.spark.ml.attribute.BinaryAttribute$$anonfun$toMetadataImpl$12.apply(attributes.scala:
	at org.apache.spark.ml.attribute.BinaryAttribute$$anonfun$toMetadataImpl$12.apply(attributes.scala:
	at scala.Option.foreach(Option.scala:257)
```

GCP needs dissection ...

- It is hard to diagnose where the issue happened
  - Running a notebook made it easier to run step by step
  - Dissecting the process helped us figure out where needed fixes
  - StringIndexer was the problem

```
In [*]: # perform one-hot encoding
from pyspark.ml.feature import OneHotEncoderModel
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoderEstimator, VectorAssembler

categorical_col = ['14','15','18','19','21','22','27','30','36','joined_column']

train_hot = train_df_impute_join

indexers = [StringIndexer(inputCol=col, outputCol="{0}_indexed".format(col)) for col in categorical_col]

encoder = OneHotEncoderEstimator(inputCols=[indexer.getOutputCol() for indexer in indexers],
                                 outputCols=["{0}_encoded".format(indexer.getOutputCol()) for indexer in indexers])
cat_assembler = VectorAssembler(inputCols=encoder.getOutputCols(),outputCol="CatFeatures")

pipeline = Pipeline(stages=indexers + [encoder, cat_assembler])
train_onehot = pipeline.fit(train_hot).transform(train_hot)
train_onehot.show()
```

```
indexed.show()
```

```
+---+-----+-----+----+-------+-----+----+----+------+---+----+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+-----
---+--------+--------+--------+--------+--------+--------+--------+--------------------+--------------------+--------------------+--------------------+--------------------+-----
----+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+-----
---+--------------------+--------------------+--------------------+--------------------+--------+
|  0|    2|    3|   4|      5|    6|   7|   8|     9| 11|  13|      14|      15|      16|      17|      18|      19|      20|      21|      22|      23|      24|      25|      26|
27|      28|      29|      30|      31|      34|      36|      37|                    |               log_2|                    |               log_3|                    |               log_4|                    |               log_5|                    |               log_6|                    lo
g_7|                    log_8|                    log_9|                   log_11|                   log_13|     log_5_imputed|      log_9_imputed|      log_2_imputed|      log_8_imputed|      log_4_imput
ed|      log_13_imputed|      log_6_imputed|      log_7_imputed|      log_3_imputed|      log_11_imputed|       joined_column|14_index|
+---+-----+-----+----+-------+-----+----+----+------+---+----+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+-----
---+--------+--------+--------+--------+--------+--------+--------+--------------------+--------------------+--------------------+--------------------+--------------------+-----
----+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+-----
---+--------------------+--------------------+--------------------+--------------------+--------+
|  0|  1.0|  5.0| 0.0| 1382.0|  4.0|15.0| 2.0| 181.0|2.0| 2.0|68fd1e64|80e26c9b|fb936136|7b4723c4|25c83c98|7e0ccccf|de7995b8|1f89b562|a73ee510|a8cd5504|b2cb9c98|37c9c164|2824a5f6|1adce
6ef|8ba8b39a|891b62e7|e5ba7672|f54016b9|07b5194c|3a171ecb|c5c50484|0.009950330853168092|  1.6114359150967734|  -4.605170185988091| 7.231294240191451|1.3887912413184778|  2.708716645645
3704|  0.6981347220709843| 5.198552278358462|  0.6981347220709843|  0.6981347220709843| 7.231294240191451| 5.198552278358462|0.009950330853168092|  0.6981347220709843|  -4.605170185988
091|  0.6981347220709843|1.3887912413184778|  2.7087166456453704|  1.6114359150967734|  0.6981347220709843|fb936136_7b4723c4...|     1.0|
|  0|  0.0| 44.0| 1.0|  102.0|  8.0| 2.0| 2.0|   4.0|1.0| 4.0|68fd1e64|f0cf0024|6f67f7e5|41274cd7|25c83c98|fe6b92e5|922afcc0|0b153874|a73ee510|2b53e5fb|4f1b46f3|623049e6|d7020589|b2847
9f6|e6c5b5cd|c92f3b61|07c540c4|b04e4670|60f6221e|3a171ecb|43f13e8b|  -4.605170185988091|        3.784416880823|0.009950330853168092| 4.625070847694428| 2.080690761080268|  0.698134722070
9843|  0.6981347220709843|1.3887912413184778|0.009950330853168092|  1.3887912413184778| 4.625070847694428|1.3887912413184778|  -4.605170185988091|  0.6981347220709843|0.009950330853168
```

```
In [*]: # perform one-hot encoding
from pyspark.ml.feature import OneHotEncoderModel
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoderEstimator, VectorAssembl

categorical_col = ['14','15','18','19','21','22','27','30','36','joined_column']

indexed = train_df_impute_join
for col in categorical_col:
    stringIndexer = StringIndexer(inputCol=col, outputCol='%s_index' % col)
    model = stringIndexer.fit(indexed)
    indexed = model.transform(indexed)
    print(col)
```

```
14
15
18
19
21
22
27
30
36
```

GCP can get expensive ...

- This is even truer if you don’t know what you are doing
- Due to the issues above, it cost us sometimes 50 dollars to get nothing out of the process
- It cost us 160 dollars for this project

Final run

	Project	Project ID	Cost
●	W261-finateam10	w261-finateam10	$18.78

Total

	Project	Project ID	Cost
●	W261-finateam10	w261-finateam10	$162.11

The 50 dollar one

	Project	Project ID	Cost
●	W261-finateam10	w261-finateam10	$41.79

GCP can be magical ...

- Finally, we were able to get it done
- It was not without tears and frustrations
- Christmastime is here
  —or as we like to call it, the most wonderful time of the year.

```
[8]: trainDF, devDF = transform_train.randomSplit([0.8,0.2], seed = 5)

[*]: accuracy, logloss, lrmodel = log_regression(trainDF , devDF)
     print('Prediction accuracy: %0.3f' %accuracy)
     print('Log loss: %0.3f' %logloss)

Prediction accuracy: 0.746
Log loss: 0.535
```