IMDB reviews classification

Alejandro Salazar Mejía, Pasang Tenzin, Jan Frąckowiak

Sample Reviews:

"This was a really interesting Halloween film. I wasnt to thrilled with the whole Thorn theory but it still makes for a good film. I liked getting to see Tommy Doyle back but sadly Donald Pleasance died right after shooting. (...)"

“comeundone, I love you! I could not have come to a better conclusion than you did about this movie and it\'s ending. My family has not seen this movie yet, but I know them too well; they will hate it.(...)"

Project Outline

Objective: Accurate classification the sentiment of reviews: positive or negative

Data: 25k reviews for training, 25k reviews for testing

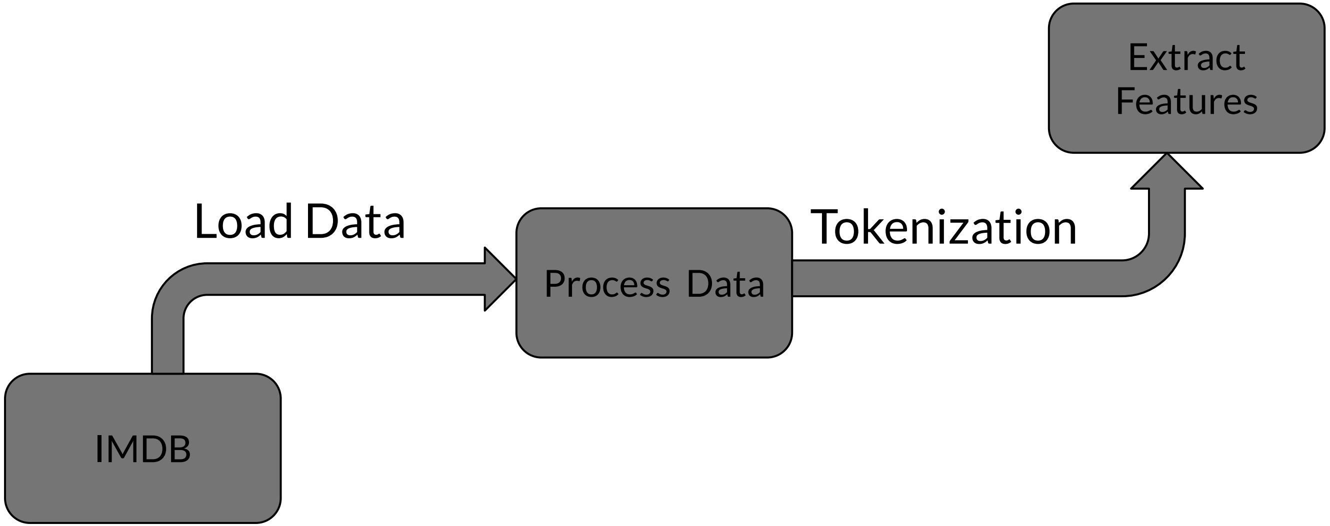
Methods: AutoAI, Manually trained models basing on embeddings

Metrics: Accuracy, Confusion Matrix, ROC-AUC

Github: https://github.com/thually/cloudTech-Summer23-24

Benchmark Model

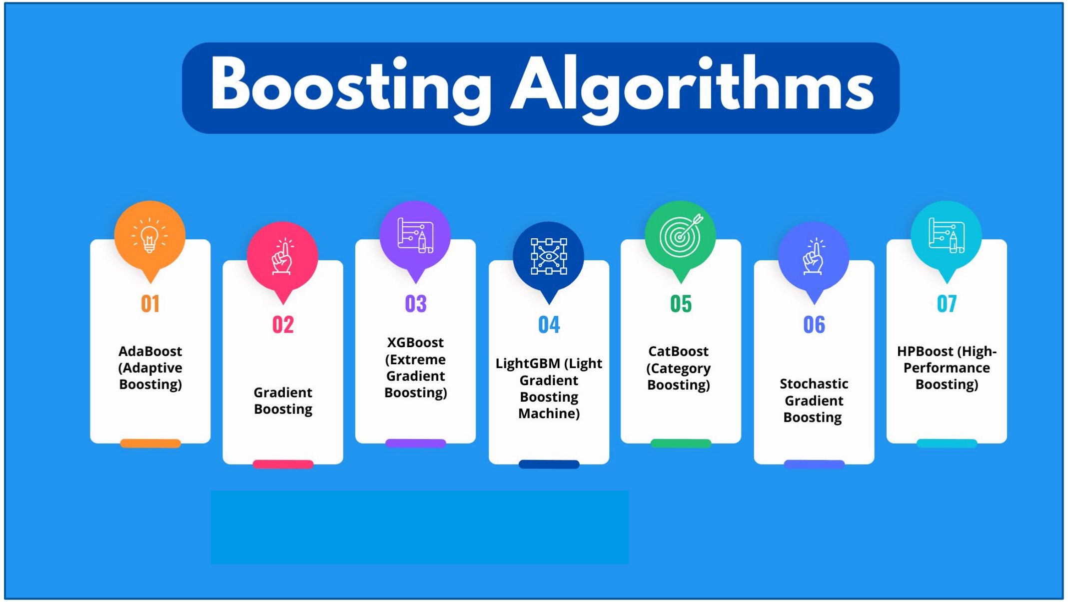
Download IMDB from IBM Cloud Object Storage



Sentiment Analysis

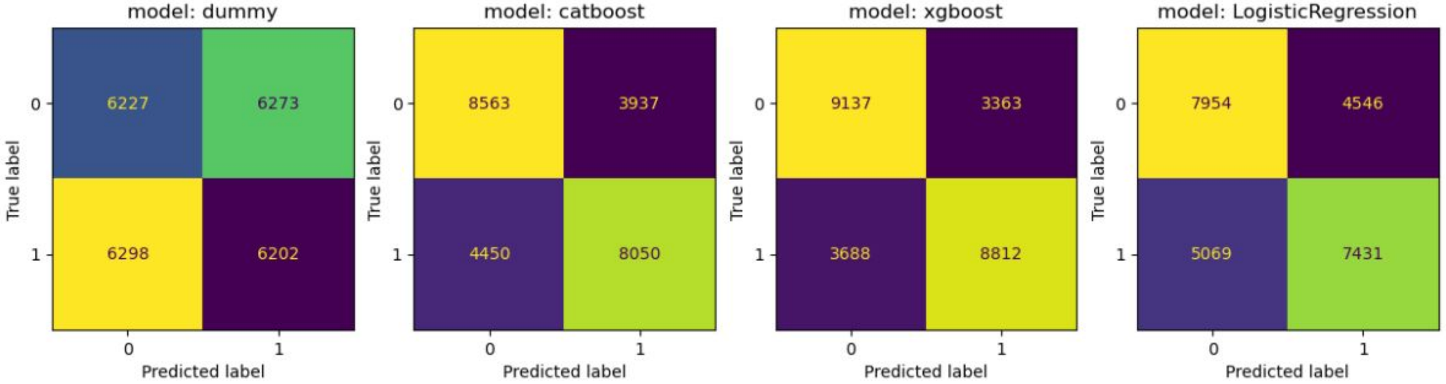
1. Load and process data.
2. Create dataframe and normalized.
3. Sampled the data and visualized.
4. Extract features.

Custom tokenization



Result with TfidfVectorizer and light normalization

```
model=dummy, accuracy=0.49716
model=catboost, accuracy=0.66452
model=xgboost, accuracy=0.71796
model=LogisticRegression, accuracy=0.6154
```



XGBoost performed best with accuracy approximately 71.8%

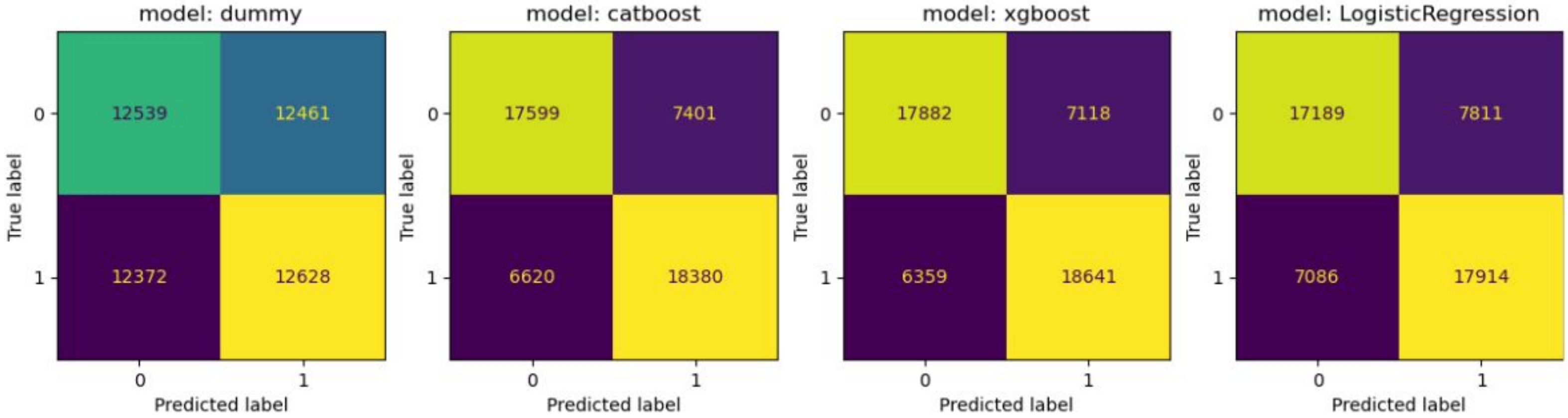
Improvement in the pre-processing of the text

1. Lowercasing all text.
2. Removing URLs.
3. Replace “
” with space.
4. Remove punctuation marks.
5. Tokenize text into words.
6. Remove non-alphabetic words.
7. Remove stop words except for “not”

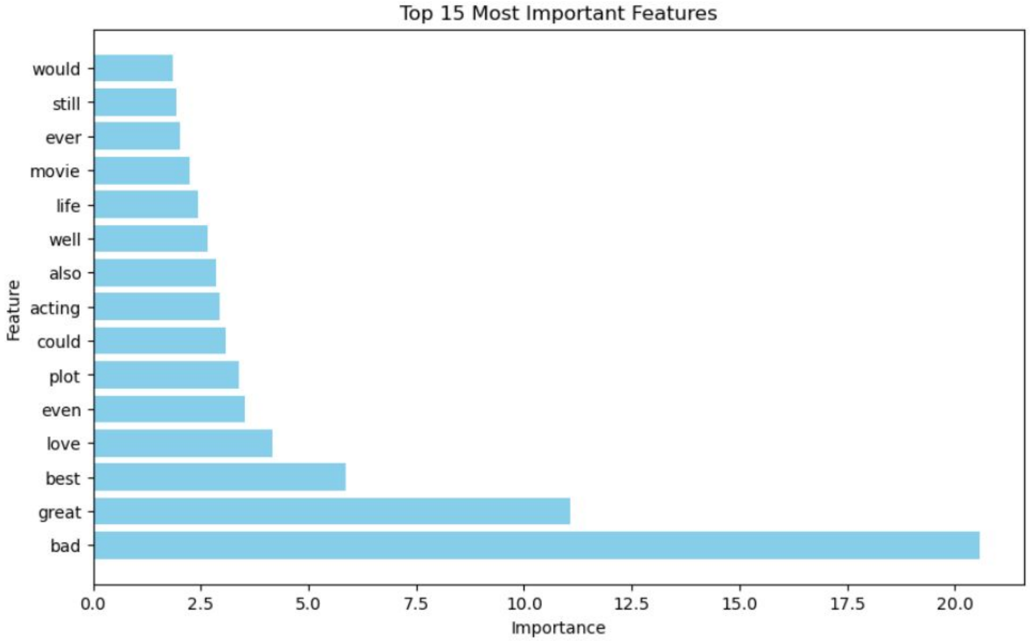
```
{'re', 'does', 'couldn', 'hasn', 'on', "you'd", 'doesn', 'there', 'myself', 'have', 'about', 'some', 'a', 'weren', 'aren', 'a
nd', 'if', 'wasn', 'haven', 'hers', 'o', 'll', 'that', 'don', 'whom', 'in', 'up', 'here', 'most', "won't", 'i', "hadn't", "yo
u've", 'didn', "didn't", 'had', 'ma', 'for', 'she', 'ourselves', "wasn't", "doesn't", 'the', "hasn't", 'with', 'when', "tha
t'll", 'is', 'now', 'an', 'of', 't', 'mightn', 'needn', 'ain', "wouldn't", 'herself', 'it', 'itself', 'he', "it's", 'will',
'out', 'once', 'my', "mustn't", 'by', 'through', 'before', 'because', 'off', 'ours', 'having', 'between', 'only', 'd', 'wher
e', 'or', "shouldn't", 'hadn', "isn't", 'any', 'isn', 'these', 'few', "aren't", 'then', 'themselves', "haven't", "weren't",
'our', "you'll", 'until', 'this', 'were', 's', 'yours', 'but', 'at', 'after', 'should', "shan't", 'your', 'against', 'y', 'ar
e', 'm', 'them', 'mustn', 'was', 'being', 'nor', 'did', 'what', "couldn't", 'down', 'can', 'won', 'why', 'theirs', 'all', 'do
ing', 'as', 'from', 'while', 'other', 'to', 'yourselves', 'been', 'shan', 'wouldn', 'you', 'more', 'yourself', 'her', 'they',
'am', 'which', 'we', 'above', 'just', 've', 'be', "you're", "mightn't", 'how', 'both', "don't", 'into', 'no', "she's", 'its',
'same', 'during', "should've", 'him', 'such', 'very', 'those', 'again', 'who', 'shouldn', 'their', 'under', 'further', 'belo
w', 'too', 'himself', 'over', 'his', 'each', 'do', 'so', 'own', "needn't", 'me', 'than', 'has'}
```

Results with our pre-processing

```
model=dummy, accuracy=0.50334
model=catboost, accuracy=0.71958
model=xgboost, accuracy=0.73046
model=LogisticRegression, accuracy=0.70206
```



Feature importance of CATboost



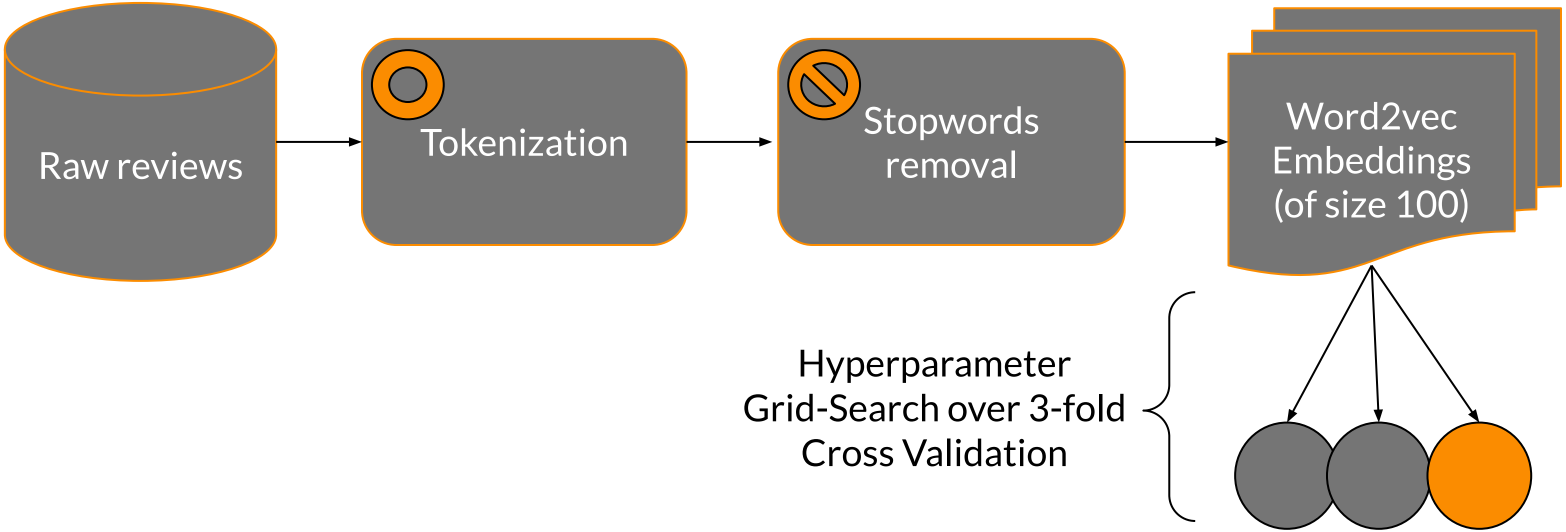

```
Accuracy: 0.71958
Precision: 0.7129281253636399
Recall: 0.7352
F1-score: 0.7238927945491423
```

Pipeline with embeddings

Embeddings Pipeline Training

Train-test split: 25k to 25k reviews (perfectly balanced)

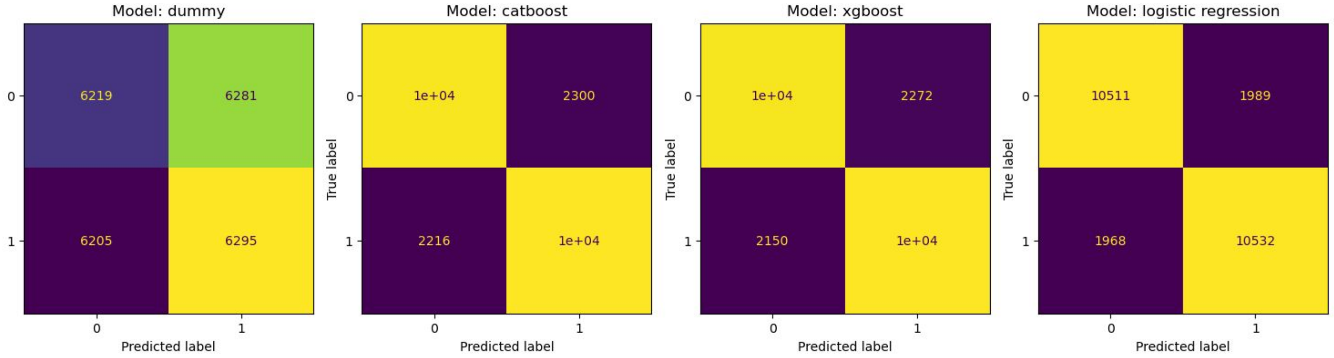
Models: Dummy, XGBoost, CatBoost, LogisticRegression



Embeddings Pipeline Results

	Dummy	Catboost	XGBoost	**LogisticReg.**
Tested Params:	1	32	48	**5**
CV Accuracy:	50.2%	91.8%	100%	**84.4%**
Test Accuracy:	49.6%	81.9%	82.3%	**84.2%**

Test C-Matrix:



AutoAI

No *Text feature engineering*

	Rank	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 8	Extra Trees Classifier	0.505	HPO-1 FE HPO-2	00:04:37
	2	Pipeline 6	Extra Trees Classifier	0.504	HPO-1	00:01:39
	3	Pipeline 4	Decision Tree Classifier	0.504		
	4	Pipeline 7	Extra Trees Classifier	0.504		
	5	Pipeline 2	Decision Tree Classifier	0.503		
	6	Pipeline 3	Decision Tree Classifier	0.502		
	7	Pipeline 1				

	labels	texts
17797	1	A film that deserved theatrical release. This ...
6458	0	What a drawn out painful experience. <br ...
14478	1	Every American who thinks he or she understand...
14825	1	I think a person would be well-advised to read...
7523	0	this is horrible film. it is past dumb. first,...
		...

Text feature engineering

When enabled, columns detected as text will be transformed into vectors to better analyze semantic similarity between strings. Enabling this setting may increase run time.

Use text feature engineering

With *Text feature engineering* - 1/2

	Rank ↑	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 6	XGB Classifier	0.722	TFE HPO-1	00:01:35
	2	Pipeline 8	XGB Classifier	0.721	TFE HPO-1 FE HPO-2	00:05:47
	3	Pipeline 2	LGBM Classifier	0.721	TFE HPO-1	00:02:34
	4	Pipeline 1	LGBM Classifier	0.721		
	5	Pipeline 4	LGBM Classifier	0.720		
	6	Pipeline 5	XGB Classifier	0.719		
	7	Pipeline 7	XGB Classifier	0.718		

Text feature engineering

When enabled, columns detected as text will be transformed into vectors to better analyze semantic similarity between strings. Enabling this setting may increase run time.

Use text feature engineering

Text feature engineering columns

- Automatically select text columns
- Manually specify text colum

Number of vectors per column

20

Specify how many vectors to create for each column during text feature engineering. A lower number is faster and a higher number is more accurate but slower.

With *Text feature engineering* - 2/2

	Rank	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 4	Logistic Regression	0.792	TFE HPO-1 FE HPO-2	00:10:56
	2	Pipeline 3	Logistic Regression	0.792	TFE HPO-1 FE	00:06:10
	3	Pipeline 8	LGBM Classifier	0.791	TFE HPO-1 FE HPO-2	00:29:29
	4	Pipeline 7	LGBM Classifier	0.791		
	5	Pipeline 2	Logistic Regression	0.791		
	6	Pipeline 1	Logistic Regression	0.790		
	7	Pipeline 6	LGBM Classifier	0.789		

Text feature engineering

When enabled, columns detected as text will be transformed into vectors to better analyze semantic similarity between strings. Enabling this setting may increase run time.

Use text feature engineering

Text feature engineering columns

- (●) Automatically select text columns
- ( ) Manually specify text columns

Number of vectors per column

30 – +

Bag of words dataframe

	Rank	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	**Pipeline 2**	XGB Classifier	**0.707**	HPO-1	00:01:01
	2	**Pipeline 1**	XGB Classifier	**0.706**	*None*	00:00:07
	3	**Pipeline 6**	LGBM Classifier	**0.705**	HPO-1	00:04:28
	4	**Pipeline 5**	LGBM Classifier	**0.704**		
	5	**Pipeline 4**	XGB Classifier	**0.704**		
	6	**Pipeline 3**	XGB Classifier	**0.704**		

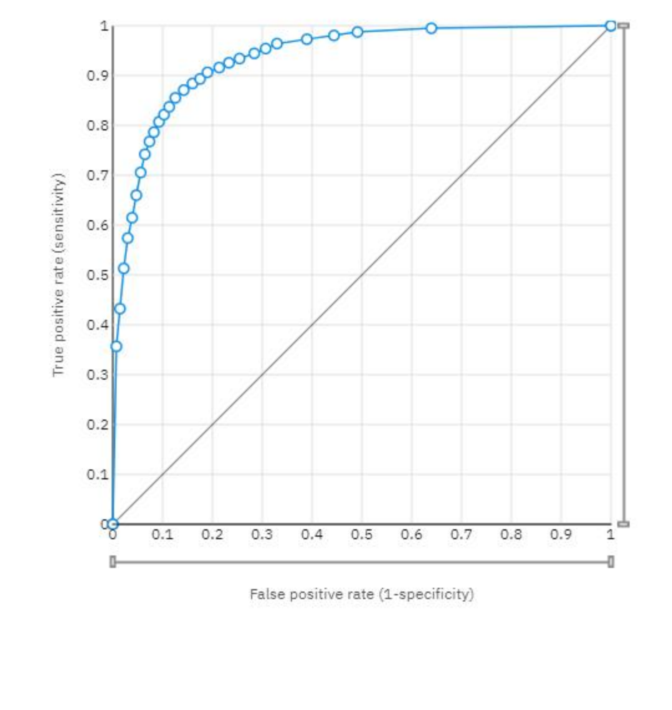
	movie	film	not	one	like	good	would	even	time	really	...
31809	0	1	1	1	1	0	0	0	0	1	...
34702	1	1	1	1	1	1	0	1	1	1	...
14222	0	1	1	1	0	0	0	1	0	0	...
26294	0	1	1	0	1	1	1	0	1	1	...
9876	0	1	1	1	1	1	1	0	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...

Word2Vec dataframe

	Rank ↑	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 2	Snap Logistic Regression	0.859	HPO-1	00:00:28
	2	Pipeline 1	Snap Logistic Regression	0.859	*None*	00:00:08
	3	Pipeline 6	LGBM Classifier	0.858	HPO-1	00:04:08
	4	Pipeline 4	Snap Logistic Regression	0.858	HPO-1 FE HPO-2	00:02:20
	5	Pipeline 3				
	6	Pipeline 8				
	7	Pipeline 7				

	vec_0	vec_1	vec_2	vec_3	vec_4	vec_5	vec_6	vec_7	vec_8	vec_9	...
0	0.183350	0.534415	0.015735	1.190066	0.360685	-0.092301	-0.194228	0.796413	-0.571112	-0.318000	...
1	0.300997	0.792131	0.077491	0.730346	0.437014	-0.205979	-0.367422	0.930643	-0.929578	-0.209355	...
2	0.331114	0.636385	0.212770	1.219322	0.521238	-0.039456	0.028477	0.918003	-1.031215	-0.089852	...
3	-0.237428	0.462750	-0.096233	0.946653	0.449601	-0.411498	-0.119159	0.645685	-0.505424	-0.195865	...
4	0.046638	0.667927	-0.056968	0.721459	0.439914	-0.214095	-0.205593	0.890287	-0.378080	-0.442387	...
...	...	...	...	...	...	...	...	...	...	...	...

P2 Snap Logistic Regression



Observed	Predicted		
	1	0	Percent correct
1	2196	304	87.8%
0	386	2114	84.6%
Percent correct	85.1%	87.4%	86.2%

Less correct More correct

Measures	Holdout score	Cross validation score
Accuracy	0.862	0.860
Area under ROC	0.936	0.935
Precision	0.851	0.851
Recall	0.878	0.873
F1	0.864	0.862
Average precision	0.934	0.932
Log loss	0.323	0.327

Environment definition

Large: 8 CPU and 32 GB RAM

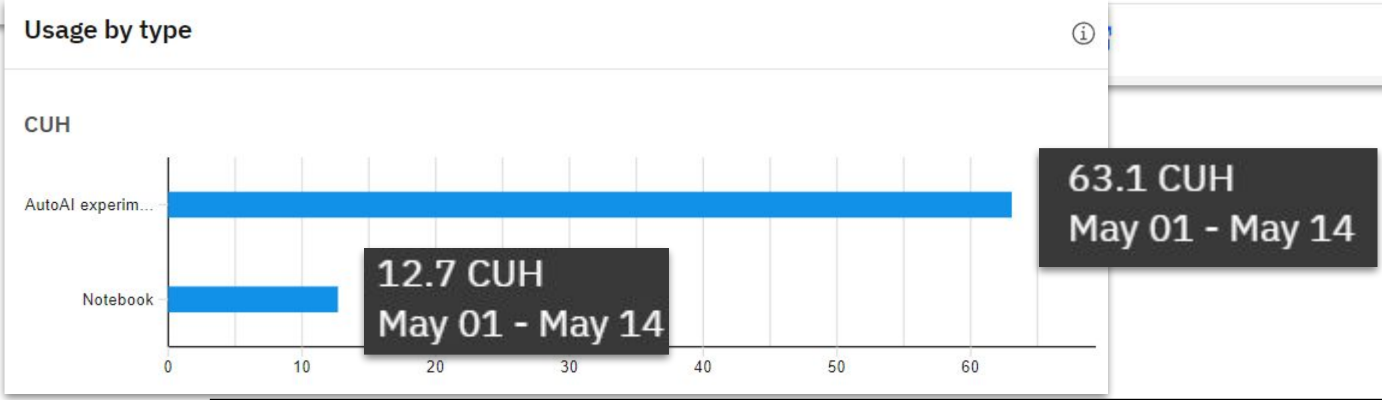
This environment definition consumes **20 capacity units per hour** for training. For details, see Watson Machine Learning plans.

Resource usage

Usage summary

For this month in this project

75.8 CUH



Let’s jump to application!