Not your parents' machine learning

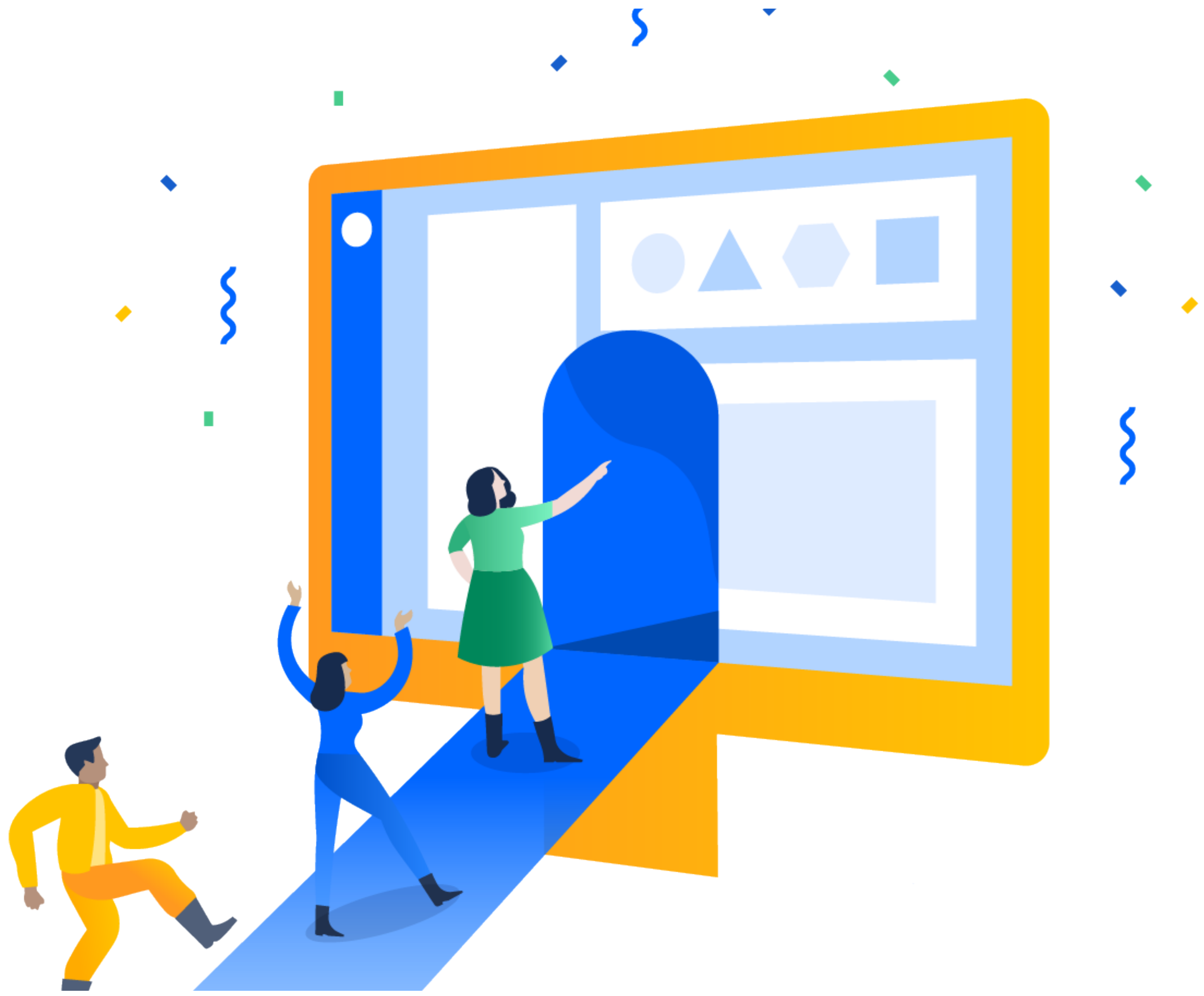
Goodman Xiaoyuan Gu | Head of Marketing Data Engineering | Atlassian

Customer churns are very costly to any business - $$$ to acquire a replacement customer

Early warnings allow us to incentivize and engage with them to improve satisfaction and retention

PROBLEM SPACE

How can we improve activation rate from evaluator -> paying customer?



USE CASES

- ***evaluator:*** who are at risk of churning but worth attempting to save? who are predicted to retain but might swing?
- ***behavior***: why those who stay and those who churn are different?
- ***content***: what content resonates with evaluators?
- ***engagement channel***: how to best engage with evaluators i.e. email, phone call, chat, push?
- ***activation rate:*** how does it change over the course of the 1st week, and what's driving it?

E2E PROCESS: CHURN PREDICTION UNLEASHED

Frame the problem

- business objective
- user and use cases
- value proposition
- assumptions

Gather status quo

- current solution
- baseline performance
- gaps and issues

Design the concept

- concept
- supervised/unsupervised/ RL
- classification / regression
- online / batch learning
- multivariate / univariate
- single machine / distributed
- design considerations:
  - timeliness
  - scale of data
  - rate of change

Define success metrics

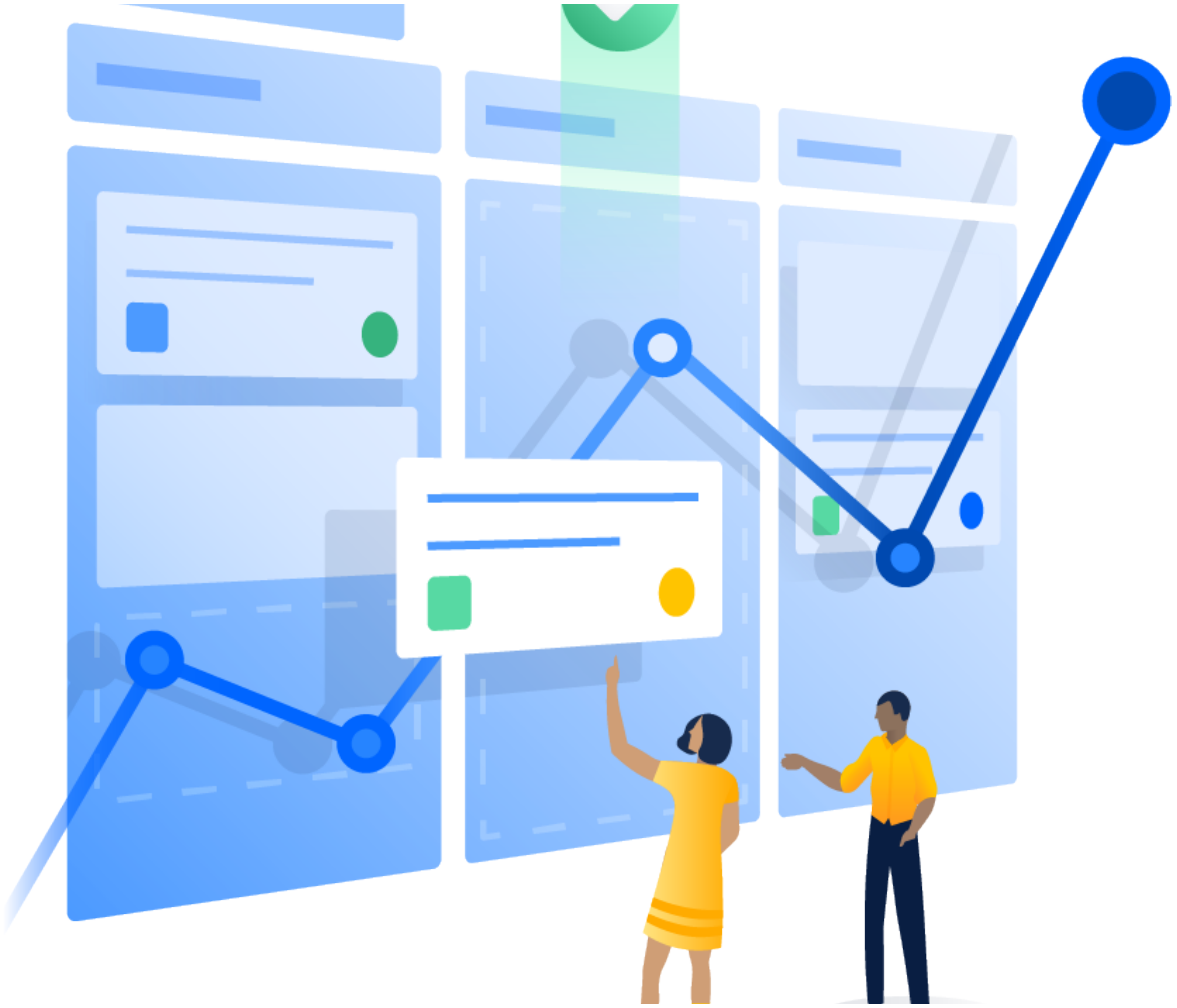
- business metric
- format: confusion matrix, classification report
- performance metrics: precision, recall, F1 score, F2 score, accuracy…

Build the solution

- collect data
- prep data for ML: wrangling, data imputing, data scaling, train/test split, cross-validation
- feature engineering: discover and visualize data to gain insights, correlation study, principal component analysis (PCA), data quality assessment, derived features development
- build and train model
- refine model and tune hyper-parameters
- evaluate model with test data
- productionize, launch and monitor

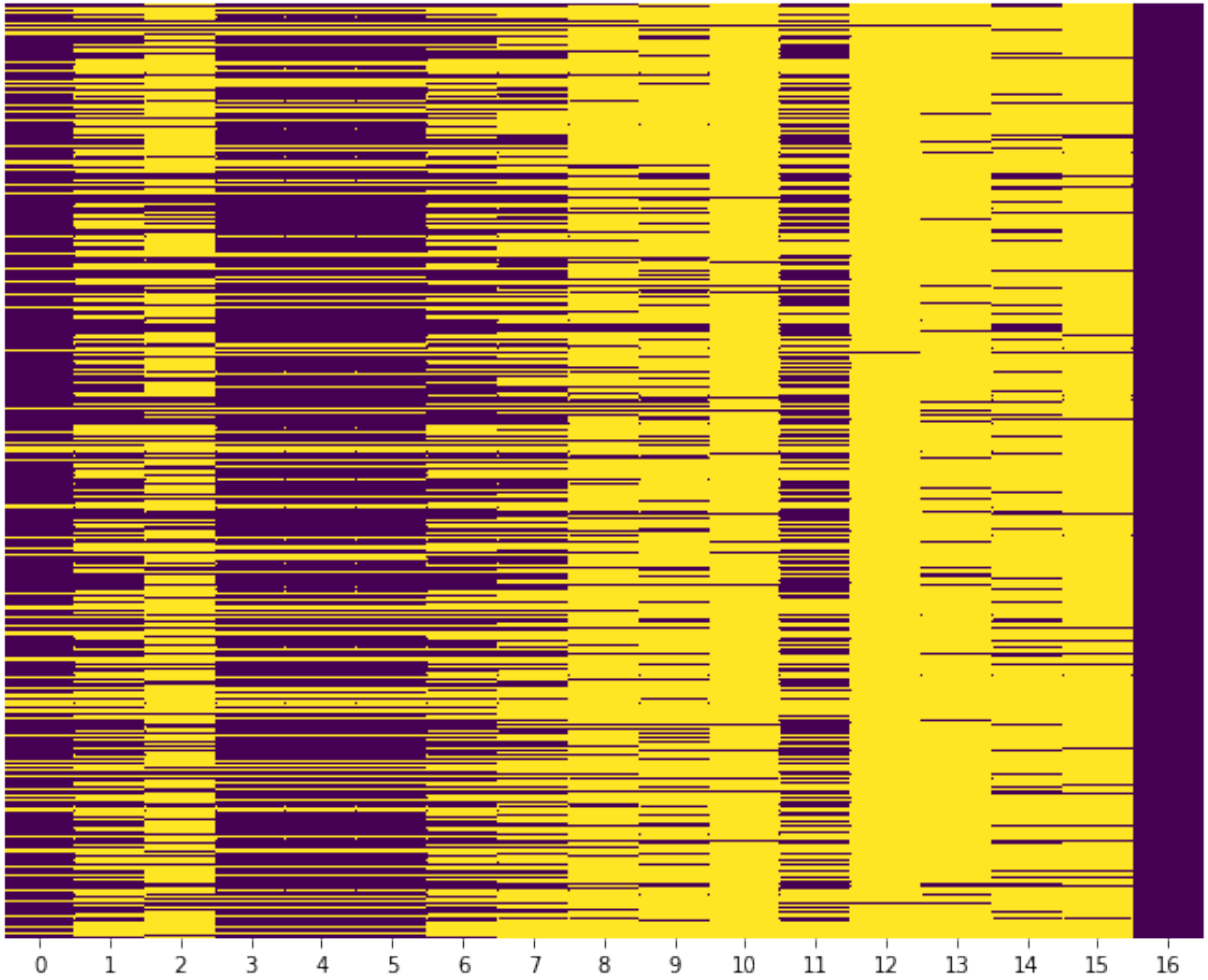
THE REAL ISSUE

Source: D. Sculley et al., "***Hidden Technical Debt in Machine Learning Systems***", in Proceedings of 28th International Conference on Neural Information Processing Systems, vol. 2, pp. 2503-2511, Montreal, Canada, Dec. 7-12, 2015



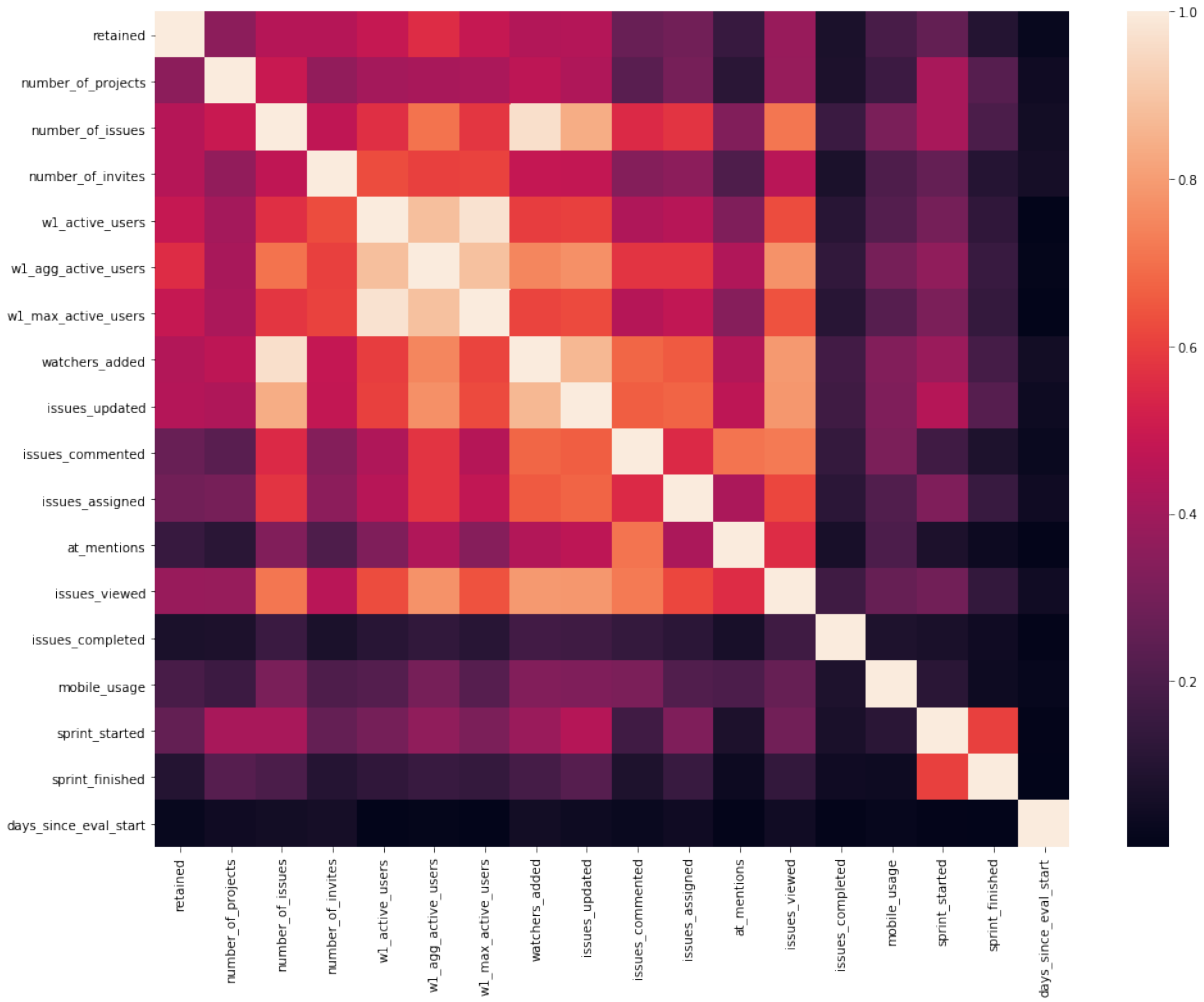
THE TRAINING DATA

- 90 days worth of product usage
- 57700 observations
- train/test split of 0.33
- data ingestion with SparkSQL jobs using EMR cluster, scheduled through Airflow
- stored on and served through AWS S3, and queryable through Athena
- re-training once/week

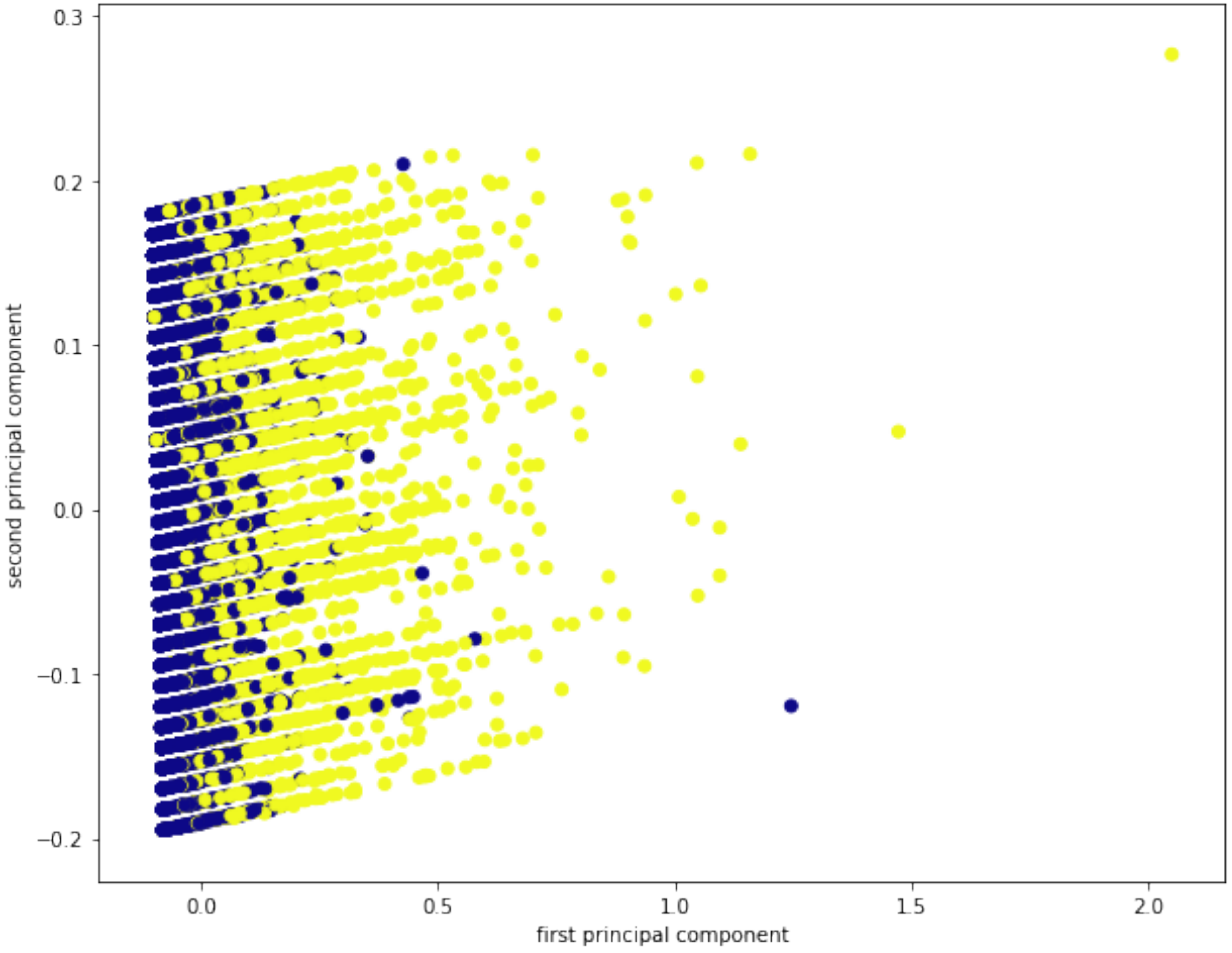


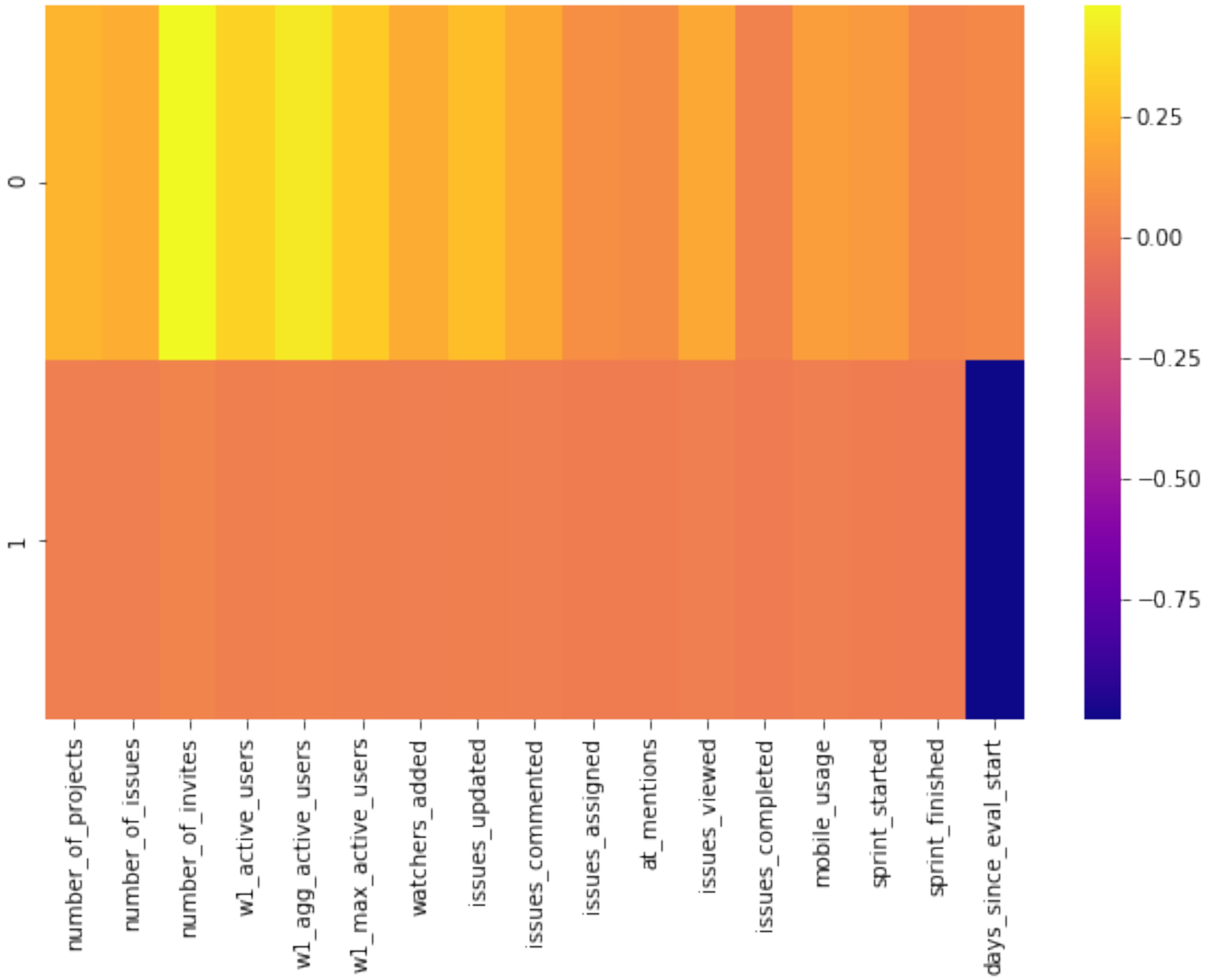
DATA PREP

```
# convert to single precision to speed up
X = dataframe_features.values.astype(np.float32)
y = dataframe_target.values.astype(np.int32)

# drop features that are extremely sparse.
drop_list = ['instance',
          'eval_start_date',
          'retained',
          ‘watchers_added’,
          ‘w1_active_users']

dataframe_features = raw_data.drop(drop_list,
axis=1, inplace=False)

# scale/normalize the data
scaler = MaxAbsScaler()
X = scaler.fit_transform(X)

# transform X to fix missing data
imputer = Imputer(strategy='median')
imputed_x = imputer.fit_transform(X)
```



w1_agg_active_users	0.56
w1_active_users	0.49
w1_max_active_users	0.49
number_of_issues	0.45
number_of_invites	0.45
issues_updated	0.44
watchers_added	0.44
issues_viewed	0.38
number_of_projects	0.35
issues_assigned	0.29
issues_commented	0.27
sprint_started	0.26
mobile_usage	0.19
at_mentions	0.15
sprint_finished	0.10
issues_completed	0.07


0.3
0.2
0.1
0.0
−0.1
−0.2
second principal component
0.0
0.5
1.0
1.5
2.0
first principal component

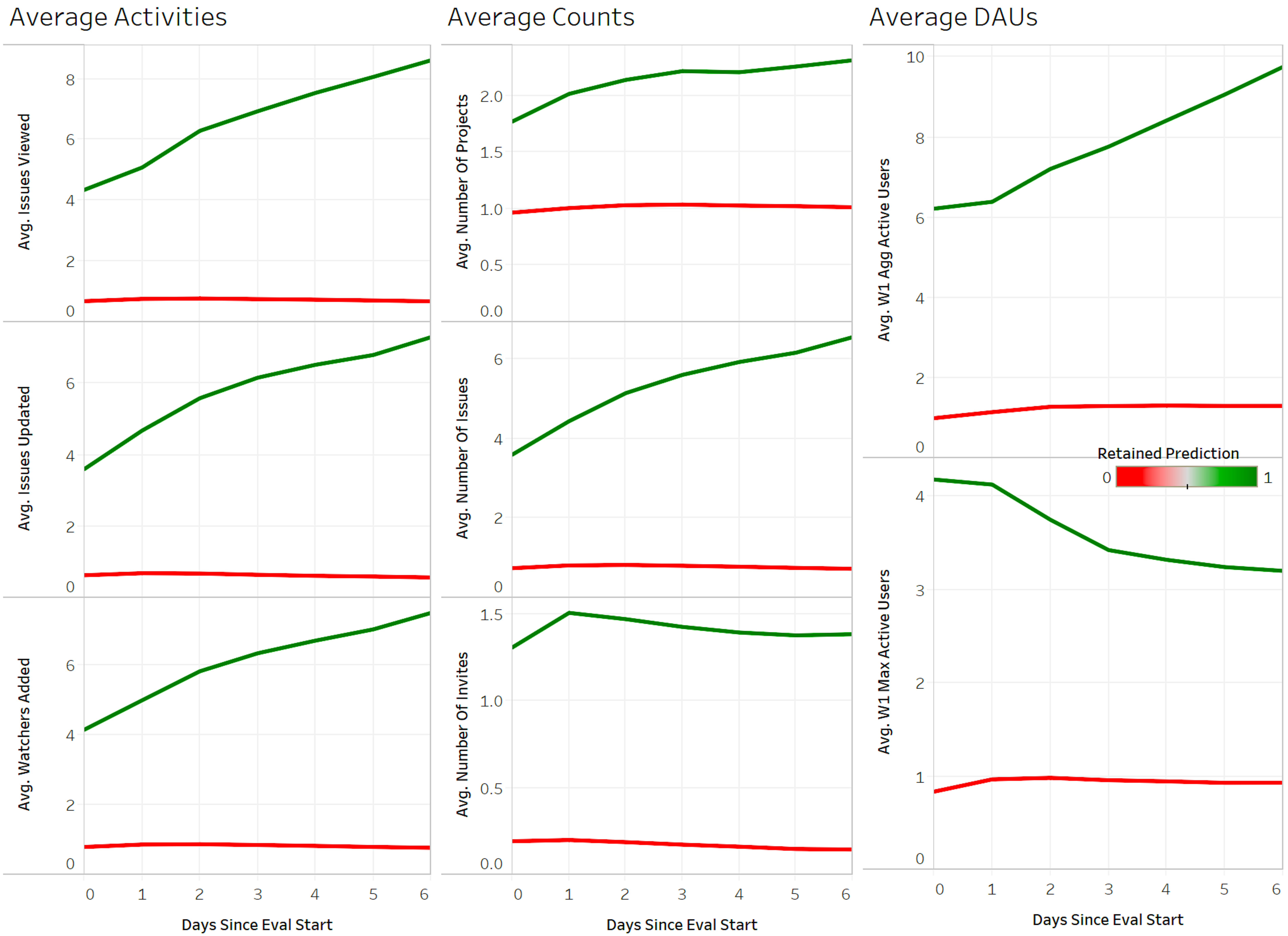

0
1
number_of_projects
number_of_issues
number_of_invites
w1_active_users
w1_agg_active_users
w1_max_active_users
watchers_added
issues_updated
issues_commented
issues_assigned
at_mentions
issues_viewed
issues_completed
mobile_usage
sprint_started
sprint_finished
days_since_eval_start
0.25
0.00
−0.25
−0.50
−0.75

```
Logistic Regression Model:
Precision score:
0.90
Recall score:
0.47
Accuracy score:
0.86
Confusion matrix:
[[14296   239]
 [ 2405  2101]]
Classification report:
             precision    recall  f1-score   support

          0       0.86      0.98      0.92     14535
          1       0.90      0.47      0.61      4506

avg / total       0.87      0.86      0.84     19041
```

```
XGBoost Model:
Precision score:
0.80
Recall score:
0.64
Accuracy score:
0.88
Confusion matrix:
[[13814   721]
 [ 1636  2870]]
Classification report:
             precision    recall  f1-score   support

          0       0.89      0.95      0.92     14535
          1       0.80      0.64      0.71      4506

avg / total       0.87      0.88      0.87     19041
```

```
Random Forest Classifier Model:
Precision score:
0.76
Recall score:
0.63
Accuracy score:
0.87
Confusion matrix:
[[13651   884]
 [ 1660  2846]]
Classification report:
             precision    recall  f1-score   support

          0       0.89      0.94      0.91     14535
          1       0.76      0.63      0.69      4506

avg / total       0.86      0.87      0.86     19041
```

```
MLP Classifier Model:
Precision score:
0.83
Recall score:
0.58
Accuracy score:
0.87
Confusion matrix:
[[14010   525]
 [ 1875  2631]]
Classification report:
             precision    recall  f1-score   support

          0       0.88      0.96      0.92     14535
          1       0.83      0.58      0.69      4506

avg / total       0.87      0.87      0.87     19041
```

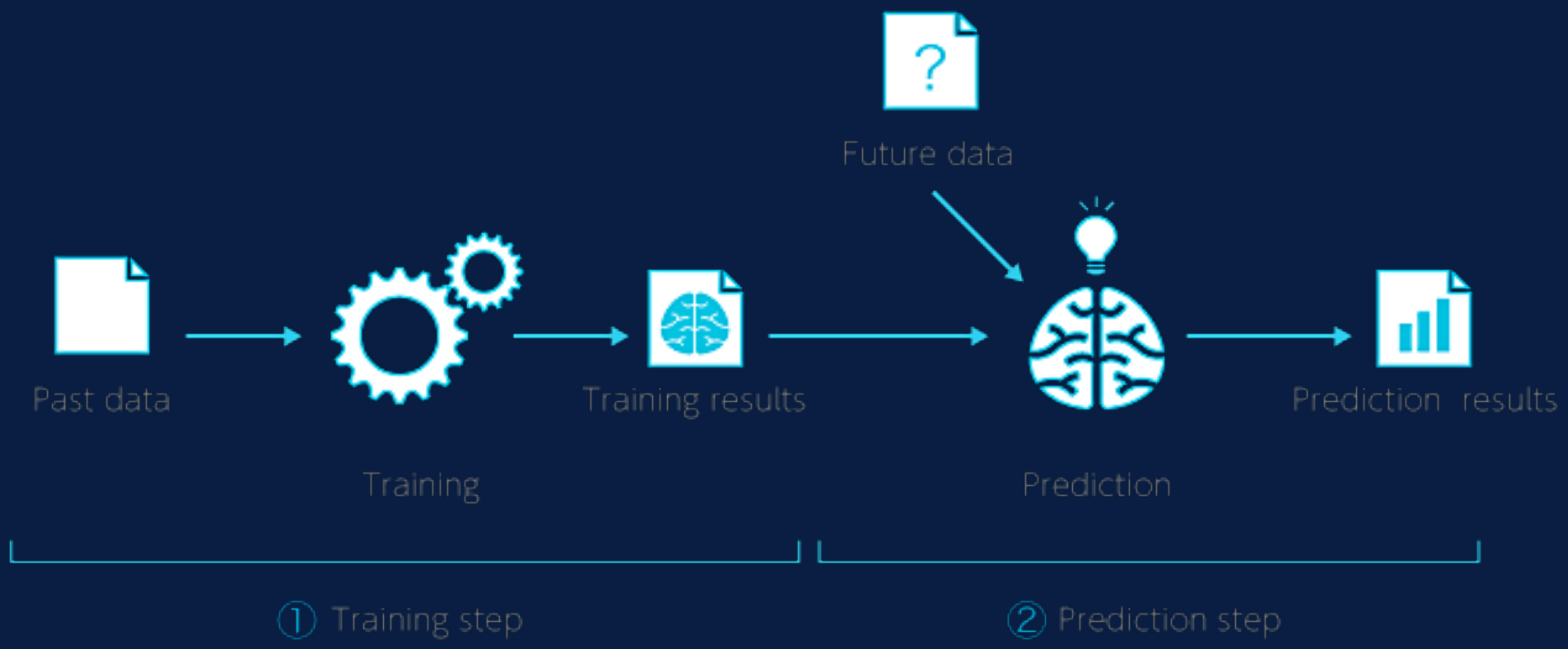

Average Activities
Average Counts
Average DAUs
Avg. Issues Viewed
Avg. Issues Updated
Avg. Watchers Added
Avg. Number Of Projects
Avg. Number Of Issues
Avg. Number Of Invites
Avg. W1 Agg Active Users
Avg. W1 Max Active Users
Retained Prediction
0
1
Days Since Eval Start
Days Since Eval Start
Days Since Eval Start

The Unreasonable Effectiveness of Data

“We may want to reconsider the tradeoff between spending time and money on algorithm development vs. spending it on corpus development”

- Michele Banko et al., Microsoft Research
- Peter Norvig et al., Google

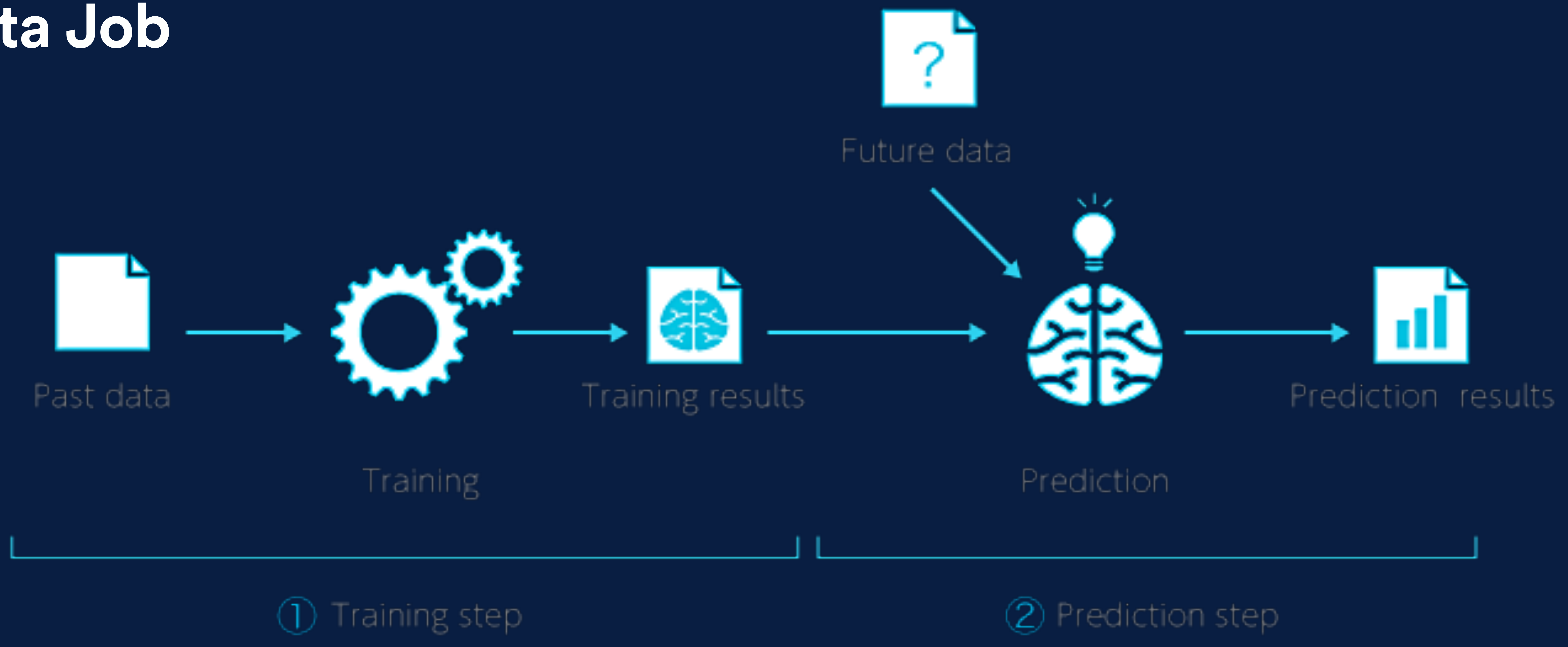
Productionizing: Training Data Schema




```
DROP TABLE IF EXISTS {marketing_schema}.instances_modeling;

CREATE EXTERNAL TABLE {marketing_schema}.instances_modeling (
  instance              INT
  ,eval_start_date        STRING
  ,retained               INT
  ,number_of_projects     INT
  ,number_of_issues       INT
  ,number_of_invites      INT
  ,w1_active_users        INT
  ,w1_agg_active_users    INT
  ,w1_max_active_users    INT
  ,watchers_added         INT
  ,issues_updated         INT
  ,issues_commented       INT
  ,issues_assigned        INT
  ,at_mentions            INT
  ,issues_viewed          INT
  ,issues_completed       INT
  ,mobile_usage           INT
  ,sprint_started         INT
  ,sprint_finished        INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://{s3_bucket_mgmt_de}/models/instances_modeling/v0'
TBLPROPERTIES ('skip.header.line.count'='1');
```

Productionizing: Training Data Job

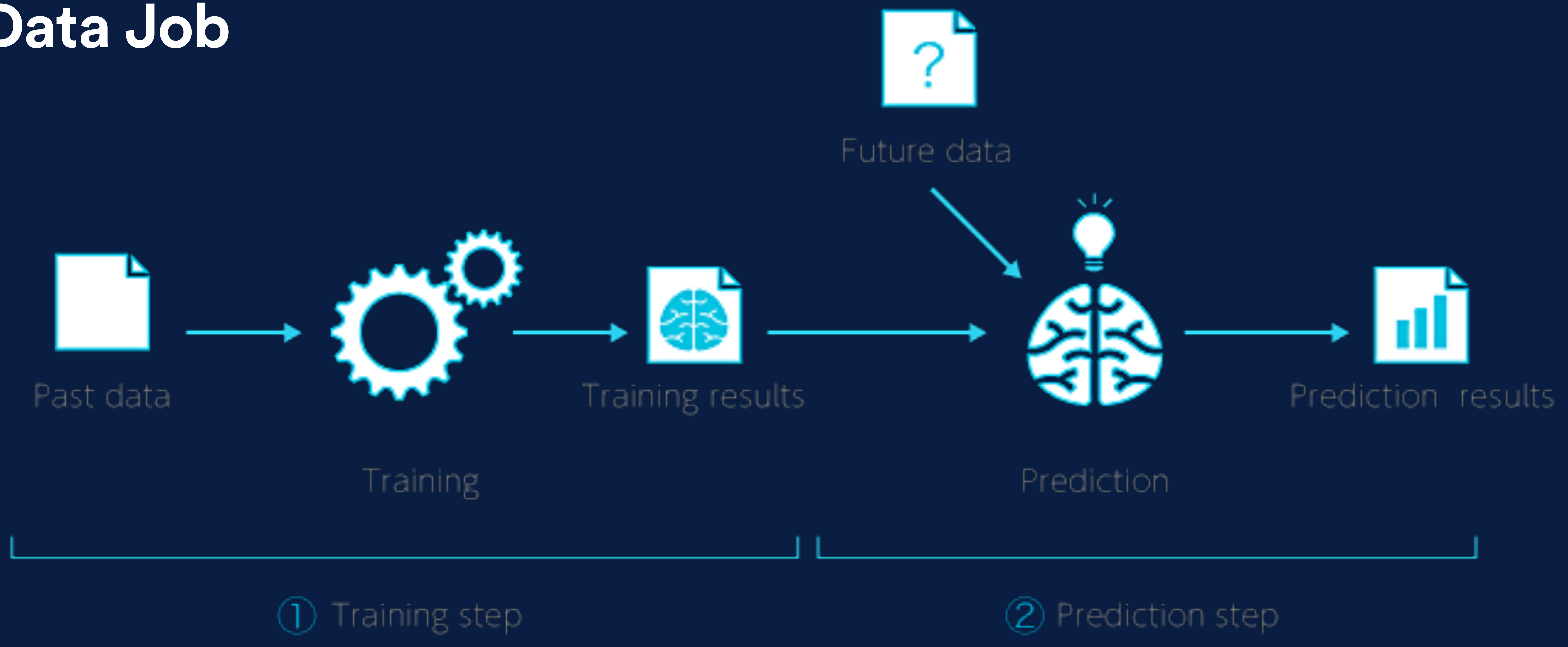
Job can be scheduled as a DAG in Airflow or entry in crontab




```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from etl_spark.util import read_text_file
import os

JOB_NAME = 'instances_modeling'
OUTPUT_S3_URI= os.path.join('s3://', S3_BUCKET_MGMT_DE, 'models',JOB_NAME,'v0')

spark = SparkSession.builder.master(spark_master).appName(JOB_NAME).enableHiveSupport().getOrCreate()

def run():
   spark.conf.set("spark.sql.parquet.binaryAsString","true")
   sql = read_text_file(os.path.join(DIR_ETL_JOBS, JOB_NAME, 'instances_modeling.sql'))
   df = spark.sql(sql.format(marketing_schema=MARKETING_SCHEMA))
   df.coalesce(1).write.csv(path=OUTPUT_S3_URI, mode='overwrite', sep=',', header=True
```

Productionizing: Prediction Data Job

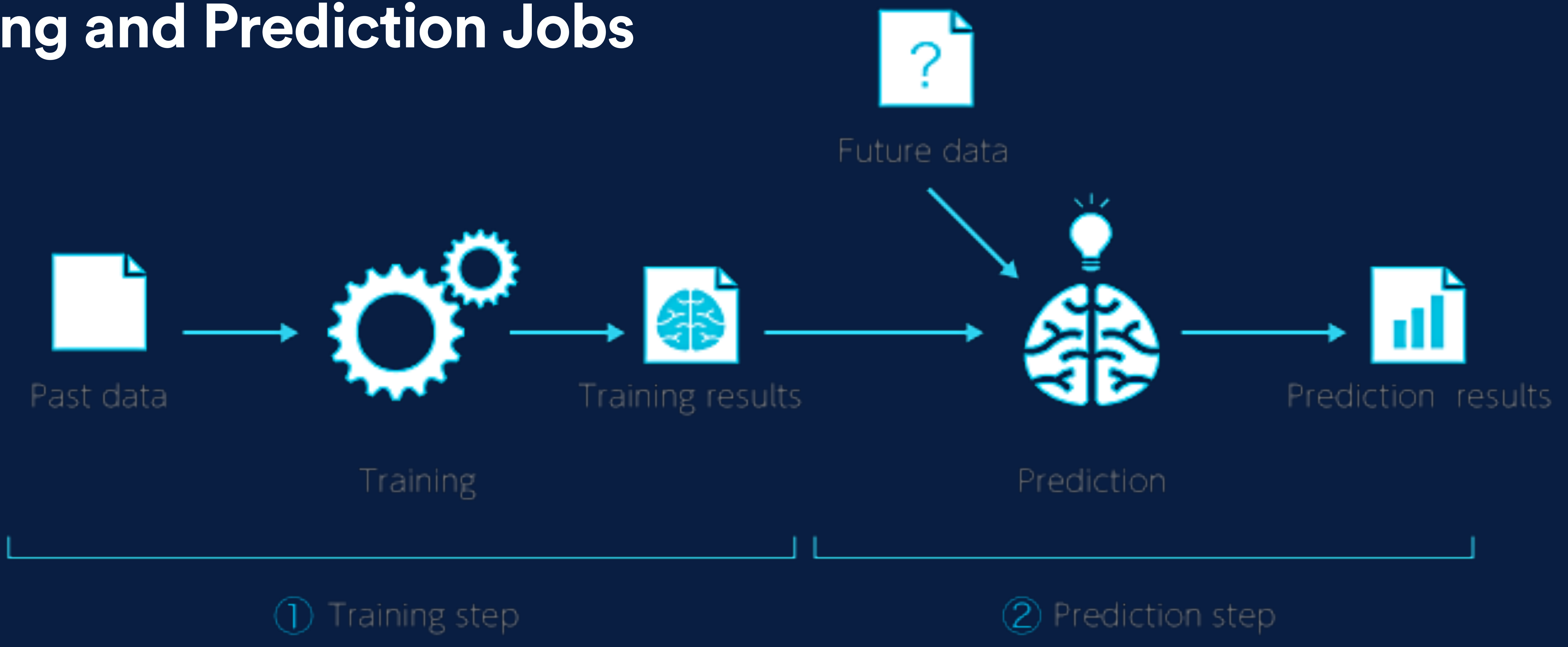
Job can be scheduled as a DAG in Airflow or entry in crontab, just more frequent




```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from etl_spark.util import read_text_file
import os

JOB_NAME = 'instances_w1'
OUTPUT_S3_URI= os.path.join('s3://', S3_BUCKET_MGMT_DE, 'models',JOB_NAME,'v0')

spark = SparkSession.builder.master(spark_master).appName(JOB_NAME).enableHiveSupport().getOrCreate()

def run():
   spark.conf.set("spark.sql.parquet.binaryAsString","true")
   sql = read_text_file(os.path.join(DIR_ETL_JOBS, JOB_NAME, 'instances_w1.sql'))
   df = spark.sql(sql.format(marketing_schema=MARKETING_SCHEMA))
   df.coalesce(1).write.csv(path=OUTPUT_S3_URI, mode='overwrite', sep=',', header=True
```

Productionizing: Model Training and Prediction Jobs

Jobs can be scheduled as a DAG in Airflow or entry in crontab on production EC2 insurance/EMR Cluster




```
#!/bin/bash

echo "start the virtual env"
export VIRTUAL_ENV_PATH=/opt/virtualenvs
PP_VENV=${VIRTUAL_ENV_PATH}/propensity-prediction-venv
source ${PP_VENV}/bin/activate

echo "run the propensity prediction model.py"
export PP_HOME=/opt/mgmt/propensity_prediction/ep
cd ${PP_HOME}
python ${PP_HOME}/model.py

echo "deactivate the virtual env"
deactivate
```

```
#!/bin/bash

echo "start the virtual env"
export VIRTUAL_ENV_PATH=/opt/virtualenvs
PP_VENV=${VIRTUAL_ENV_PATH}/propensity-prediction-venv
source ${PP_VENV}/bin/activate

echo "run the propensity prediction predict.py"
export PP_HOME=/opt/mgmt/propensity_prediction/ep
cd ${PP_HOME}
python ${PP_HOME}/predict.py

echo "deactivate the virtual env"
deactivate
```

Training

XGBoost

- Single algorithm used by ~60% Kaggle Competition winning teams
- *Extreme Gradient Boosting*
    - Sparse-aware implementation fixing missing data
    - Block Structure for parallel tree construction
    - Parallelization using CPU cores during training
    - Distributed Computing for large models
    - Out-of-Core Computing for very large datasets that don’t fit into memory
    - Cache Optimization of data structures and algorithm
    - Continued Training - boost fitted model on new data

```
...
from xgboost import XGBClassifier

# data prep and feature engineering

# with tuned hyperparameters
model = XGBClassifier(
   learning_rate=0.1,
   n_estimators=200,
   max_depth=3,
   min_child_weight = 6,
   gamma = 0,
   subsample=0.5,
   colsample_bytree=1.0,
   colsample_bylevel=1.0,
   objective='binary:logistic',
   nthread=-1,
   scale_pos_weight = 1,
   seed=27)

# train the model
model.fit(X_train, y_train)

# make predictions
predictions = model.predict(X_test)

# evaluate with test set

# persist model
joblib.dump(model, MODEL_PATH)
s3_r.meta.client.upload_file(MODEL_PATH, Bucket=BUCKET,
Key=MODEL_PATH_REMOTE)
```

Prediction

XGBoost

- Single algorithm used by ~60% Kaggle Competition winning teams
- *Extreme Gradient Boosting*
    - superior overall performance
    - excellent execution speed
    - relatively small footprint
    - easy model persistency

```
...
from xgboost import XGBClassifier

obj = s3.get_object(Bucket=BUCKET,
    Key=objs['Contents'][-1]['Key'])

# load prediction data
data_frame =
    pd.read_csv(io.BytesIO(obj['Body'].read()))

s3_model.meta.client.download_file(Bucket=
    BUCKET, Key=MODEL_PATH_REMOTE, Filename=MODEL_PATH)

# load persisted XGBoost model
predictor = joblib.load(MODEL_PATH)

#feature selection

# scale the values of selected features
scaler = MaxAbsScaler()
features_scaled = scaler.fit_transform(features_selected)

# transform features
imputer = Imputer(strategy='median')
imputed_x = imputer.fit_transform(features_selected)

# make predictions
new_predictions = predictor.predict(imputed_x)

# add predictions as a new column to the original data frame
data_frame['prediction_retained'] = new_predictions
new_data.to_csv(LOCAL_FILE_PATH, index=False)
s3_r.meta.client.upload_file(LOCAL_FILE_PATH, Bucket=BUCKET,
Key=FILE_PATH)
```

PRODUCTIONIZING

What are some challenges you can imagine?

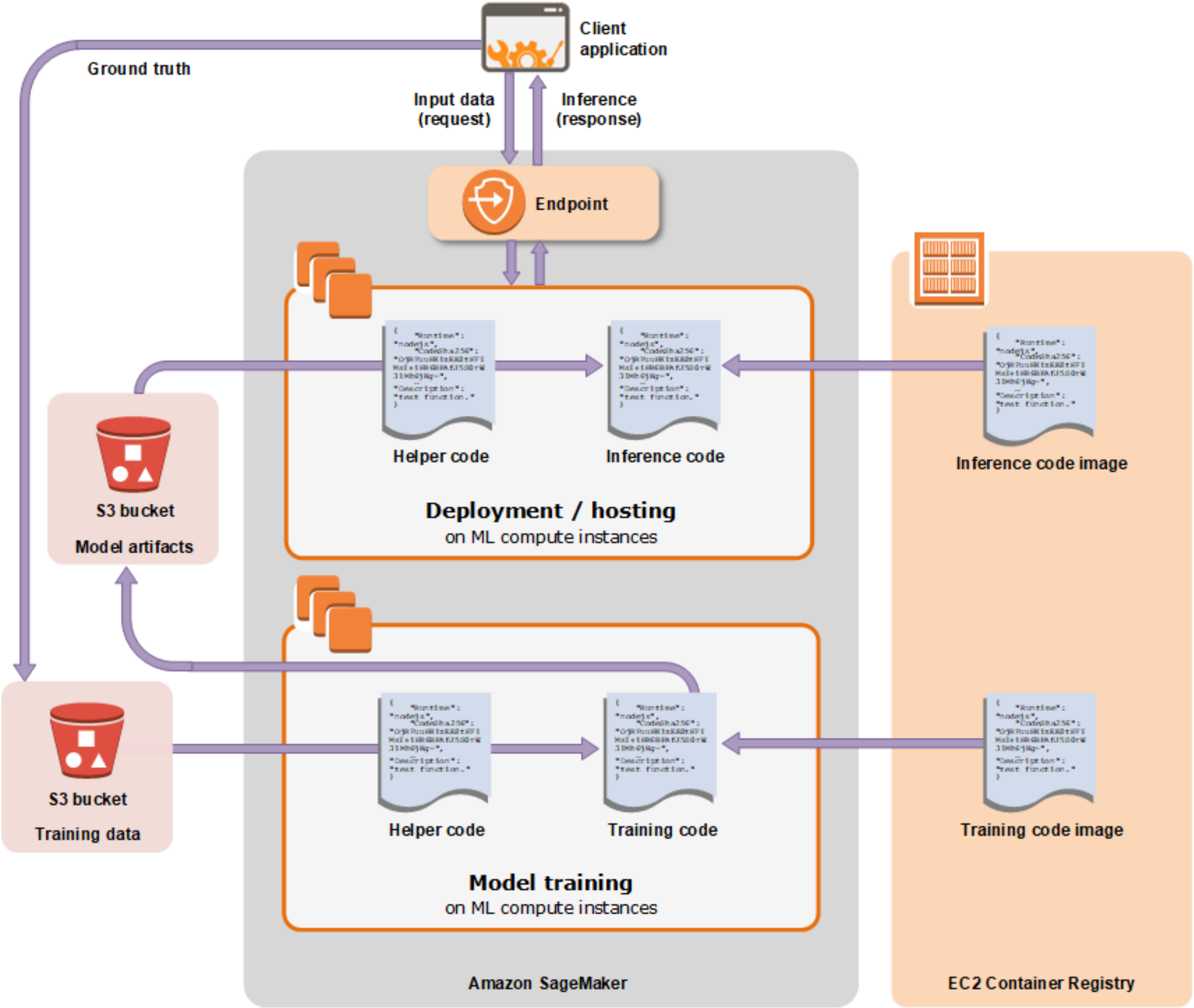
AMAZON SAGEMAKER

- managed service - easily build, train, and deploy machine learning models
- hosted Jupyter notebooks - explore and visualize training data
- 12 algorithms pre-installed and optimized
- pre-configured to run TensorFlow and Apache MXNet
- single-click training in the console or with a simple API call
- automated Hyperparameter Optimization (HPO)
- deploys model on cluster for performance and availability
- built-in A/B testing capabilities for experiments
- easy to integrate machine learning models into applications by providing an HTTPS endpoint

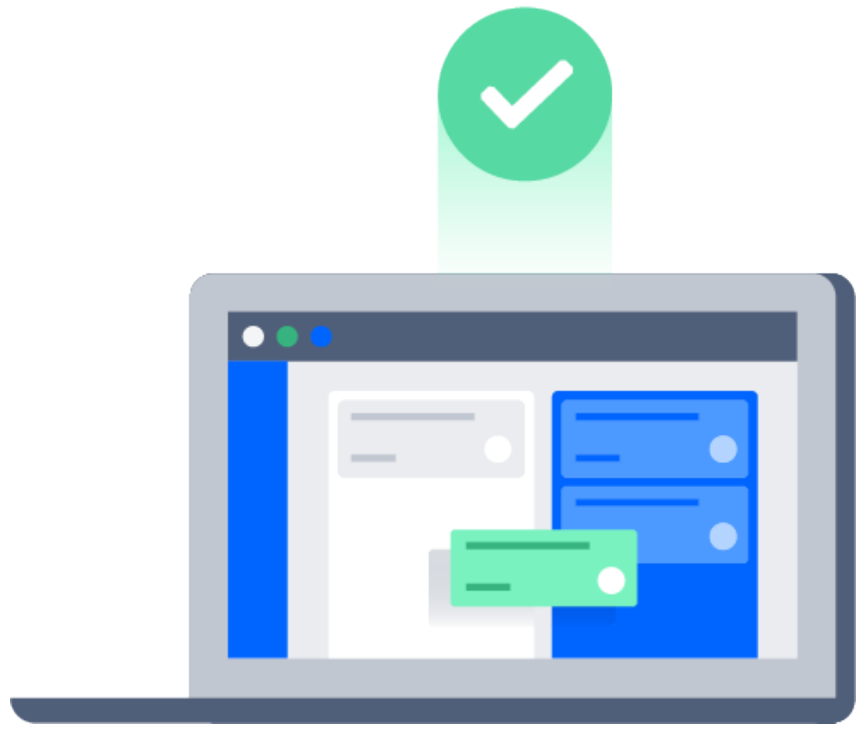
Amazon SageMaker

★complexity transparency
★faster time to market
★tight integration with existing data workflow

Workflow Demo of Churn Prediction with Sagemaker


Client application
Ground truth
Input data (request)
Inference (response)
Endpoint
Helper code
Inference code
Inference code image
S3 bucket
Model artifacts
Deployment / hosting
on ML compute instances
S3 bucket
Training data
Helper code
Training code
Training code image
Model training
on ML compute instances
Amazon SageMaker
EC2 Container Registry

We have gone through this



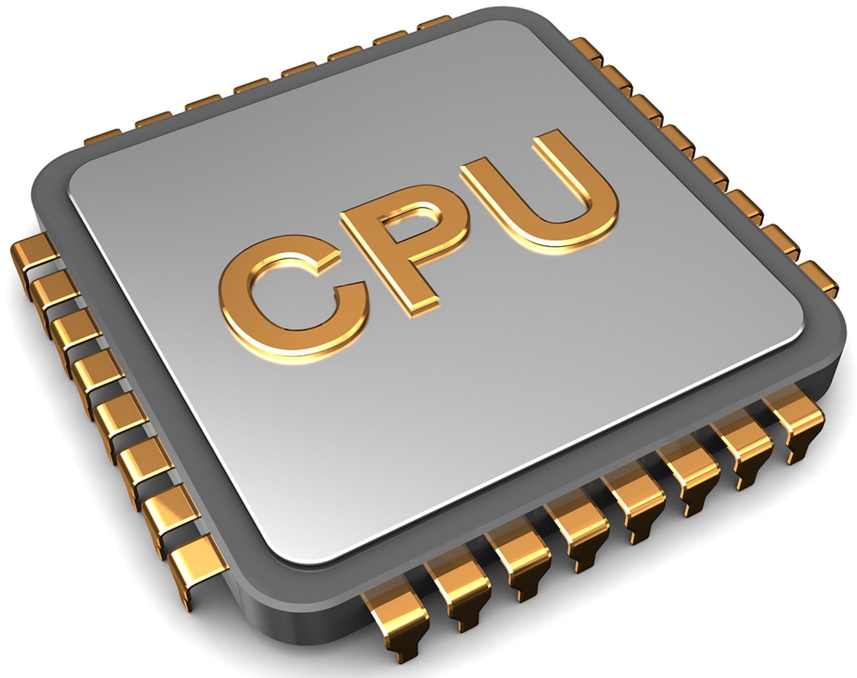
Local Machine

Cloud EC2 Instances

Amazon SageMaker

Cloud ML Platform

We will go build



Churn Prediction Unleashed

(CPU)

Generic Prediction Utility

(GPU)

Application Specific Inference Capability

(ASIC)

ATLASSIAN

We are hiring…