Chuan Xu, Jose A. Rodilla, Ting Ting Liu

Kiva Loan Time Prediction

How Kiva works

Choose a borrower

Browse categories of borrowers— people looking to grow businesses, go to school, switch to clean energy and more.

Make a loan

Select a borrower who you connect with and help fund a loan with as little as $25.

Get repaid

Receive updates on your loans and see the dollars return to your Kiva account.

Repeat!

Use the repayment to support another borrower, or withdraw your money.

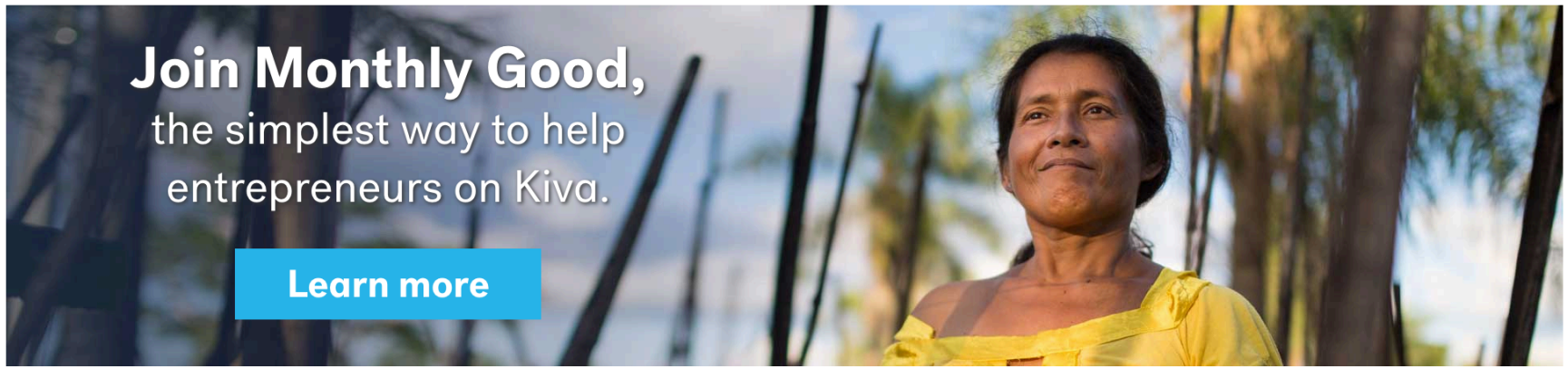
Choose a category

By country or activity

Search

Choose for me

All loans



Project Objective

To learn how Spark and Distributed Data Systems can help with our machine learning model at Kiva.

Analytics Goal

To predict how much time (in hours) a given loan will take to reach full funding.

Data
~2G

build.kiva: The Home for Kiva

build.kiva.org

Apps | Google | ML | Bookmarks | Statistics Views | Catchafire | Airtable | Math | edX | Quartz | Quora | Codecademy

Login | Register | Forum | Kiva.org

BUILD.KIVA

Home | Docs ▾ | API | Blog | My Apps

We're Open!

Developers worldwide are helping us make it easy and transparent to lend to the working poor via microfinance and the Kiva API. This is the home for all the information and tools you need to join us!

Learn

Check out our developer docs or explore the API Reference.

Build

Create the application Kiva Lenders have always dreamed of.

Share

Let the world experience microlending through the applications you've built!

Ready to Explore?

Check out our section on Getting Started or jump straight to testing out some of the API methods below.

API Spotlight

lenders/<id>/teams
Returns teams that a particular lender is a member of.
Try it: JSON | XML | HTML

teams/<ids>
Get detailed information about a lending team or a list of teams.
Try it: JSON | XML | HTML

loans/newest
This method returns all of the loans raising funds right now at Kiva, newest

API Status

OK

Build **1a5f1fe0c7d40354c6f03404cf18ed91c32689da** was released **19 hours, 44 minutes ago**.

Data Snapshots

Need to work with a lot of our data all at once? Every night we archive public data from the Kiva API into easily downloadable snapshots. Grab them in your favorite format here:

JSON Snapshot (kiva_ds_json.zip)

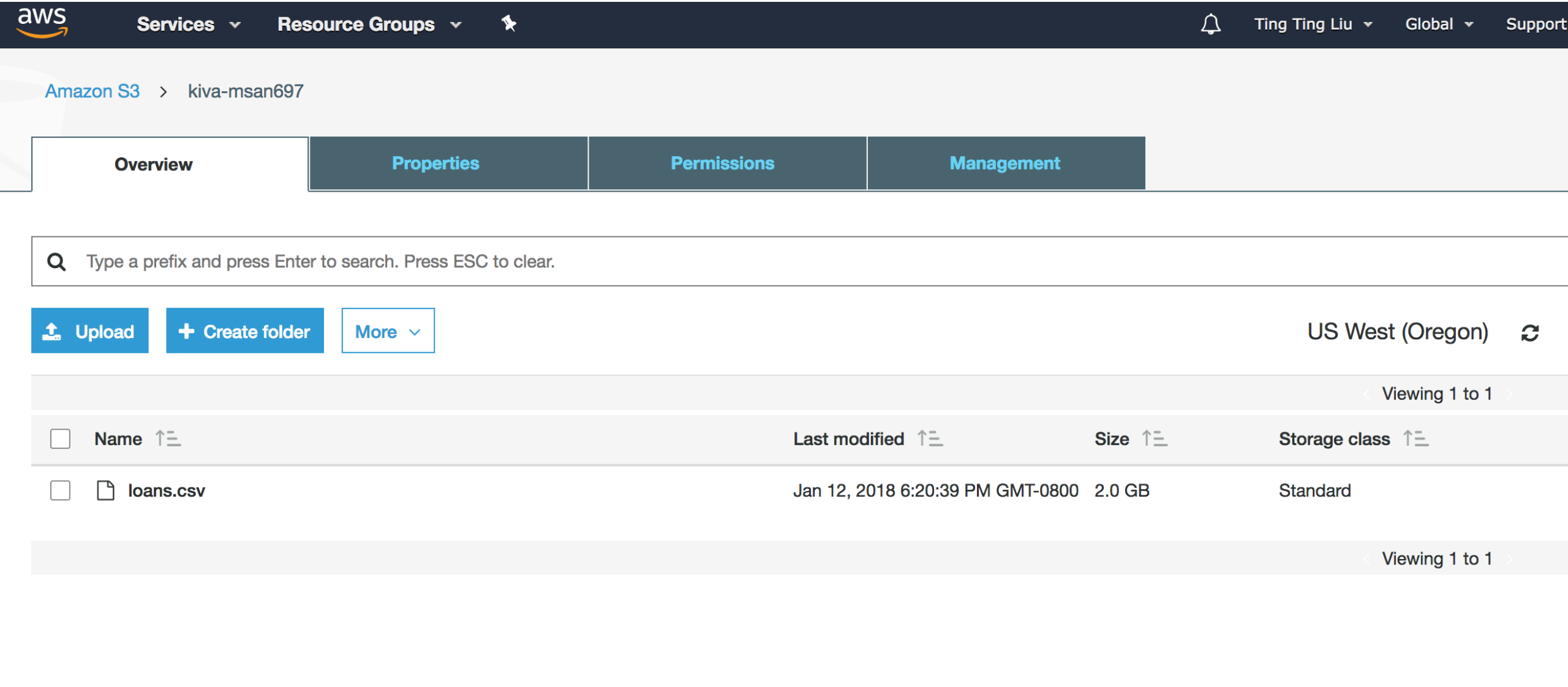
CSV Snapshot (kiva_ds_csv.zip)

More on Data Snapshots »

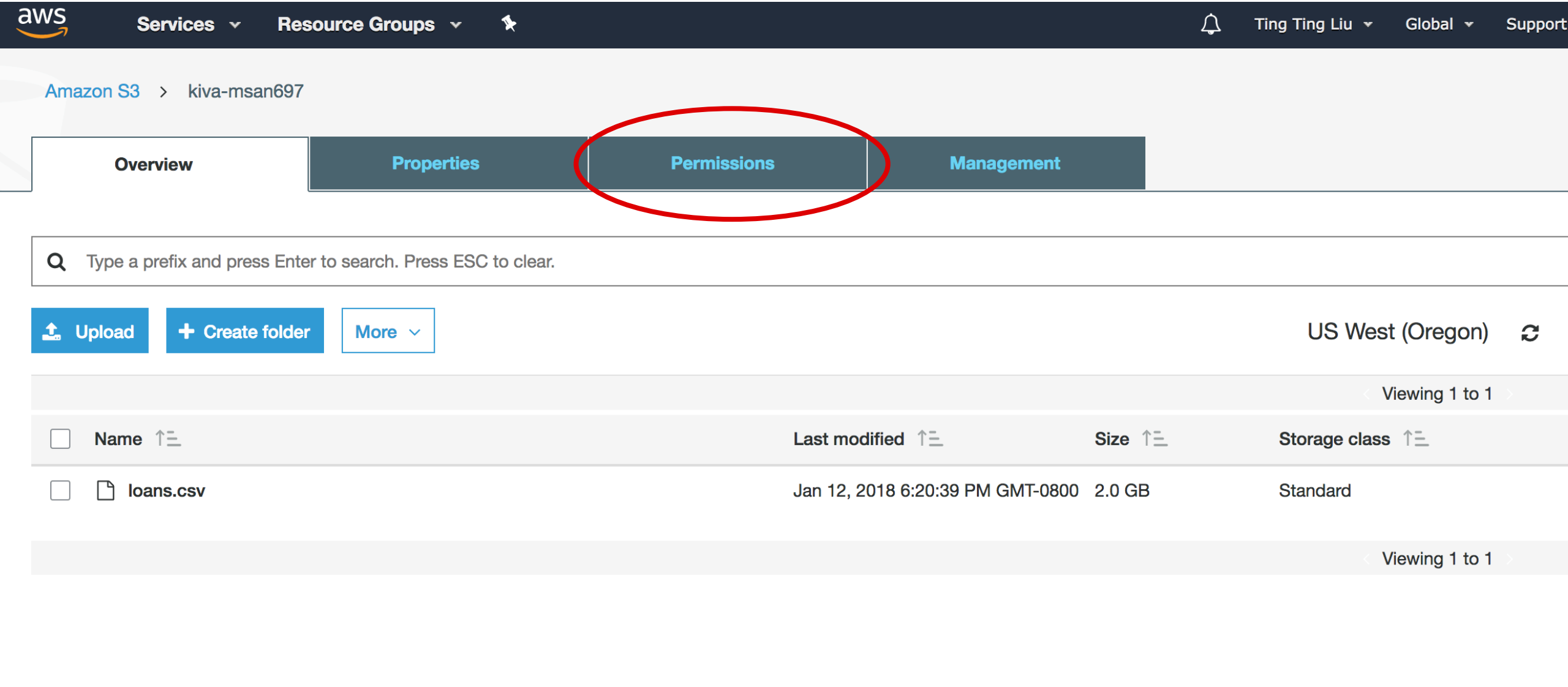
Feature	Description
loan_amount	principal of the loans (in dollars)
status	funded/expired
activity	borrower activity
sector	borrower sector
country_code	two letter country identifier
currency	country currency
posted_time	timestamp of loan posted
funded_time	timestamp of loan funded
term_in_months	repayment conditions
borrower_genders	borrower gender(s)

kiva

Data on S3


aws
Services
Resource Groups
Ting Ting Liu
Global
Support
Amazon S3 > kiva-msan697
Overview
Properties
Permissions
Management
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US West (Oregon)
Viewing 1 to 1
Name
Last modified
Size
Storage class
loans.csv
Jan 12, 2018 6:20:39 PM GMT-0800
2.0 GB
Standard
Viewing 1 to 1

Data on S3


aws
Services
Resource Groups
Ting Ting Liu
Global
Support
Amazon S3 > kiva-msan697
Overview
Properties
Permissions
Management
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US West (Oregon)
Viewing 1 to 1
Name
Last modified
Size
Storage class
loans.csv
Jan 12, 2018 6:20:39 PM GMT-0800
2.0 GB
Standard
Viewing 1 to 1

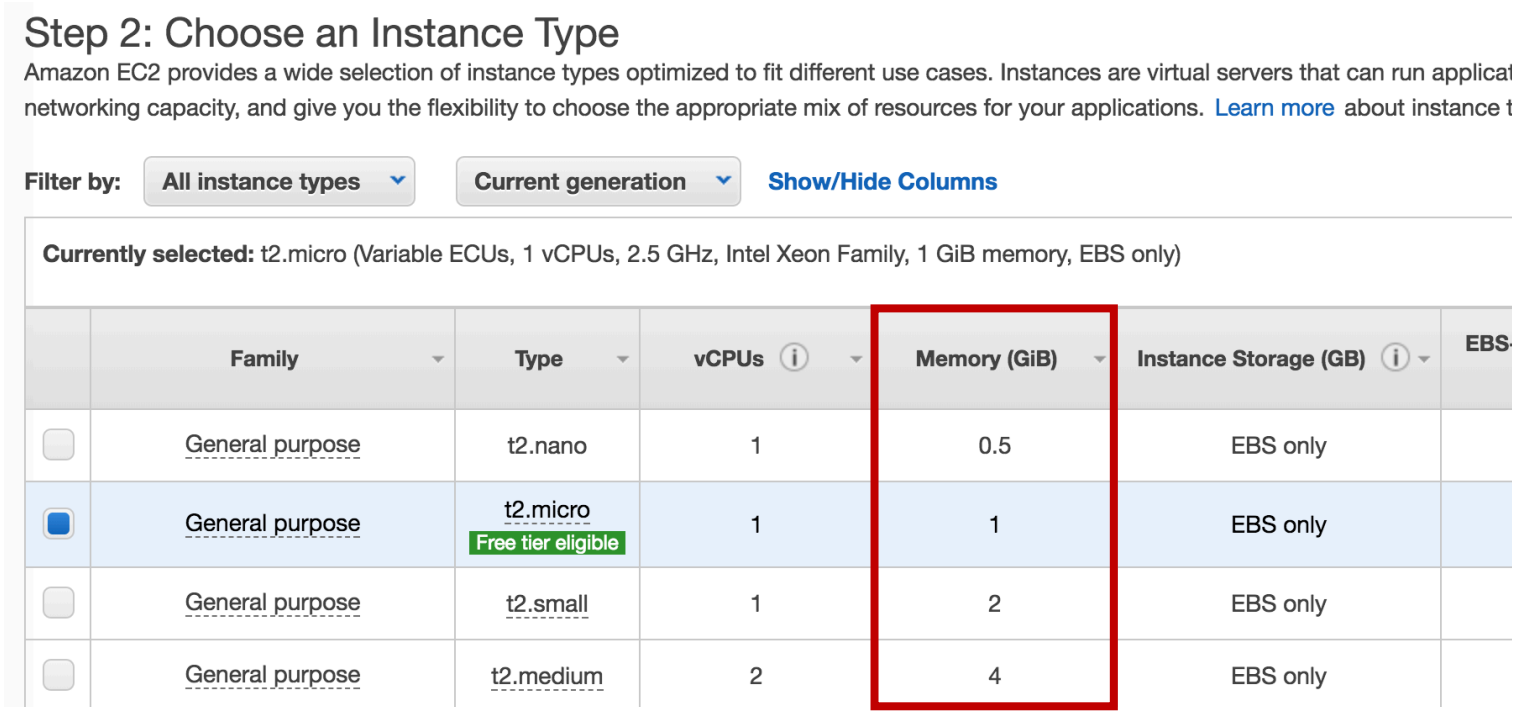
Amazon S3 → MongoDB

```
[ec2-user@ip-172-31-29-169 ~]$ source .bashrc
[ec2-user@ip-172-31-29-169 ~]$ aws s3 cp s3://kiva-msan697/loans.csv .
download: s3://kiva-msan697/loans.csv to ./loans.csv
[ec2-user@ip-172-31-29-169 ~]$ mongoimport --db loanDB --collection loans loans.csv --type=csv --headerline
2018-01-16T01:12:39.280+0000    connected to: localhost
2018-01-16T01:12:42.275+0000    [........................] loanDB.loans 49.6MB/1.98GB (2.4%)
2018-01-16T01:12:45.276+0000    [#.......................] loanDB.loans 97.6MB/1.98GB (4.8%)
2018-01-16T01:12:48.276+0000    [#.......................] loanDB.loans 146MB/1.98GB (7.2%)
2018-01-16T01:12:51.276+0000    [##......................] loanDB.loans 196MB/1.98GB (9.7%)
2018-01-16T01:12:54.276+0000    [##......................] loanDB.loans 243MB/1.98GB (12.0%)
2018-01-16T01:12:57.275+0000    [###.....................] loanDB.loans 292MB/1.98GB (14.4%)
2018-01-16T01:13:00.275+0000    [####....................] loanDB.loans 341MB/1.98GB (16.8%)
2018-01-16T01:13:03.276+0000    [####....................] loanDB.loans 388MB/1.98GB (19.1%)
2018-01-16T01:13:06.275+0000    [#####...................] loanDB.loans 435MB/1.98GB (21.5%)
2018-01-16T01:13:09.275+0000    [#####...................] loanDB.loans 465MB/1.98GB (23.0%)
2018-01-16T01:13:12.275+0000    [#####...................] loanDB.loans 474MB/1.98GB (23.4%)
2018-01-16T01:13:15.275+0000    [######..................] loanDB.loans 519MB/1.98GB (25.6%)
2018-01-16T01:13:18.275+0000    [######..................] loanDB.loans 567MB/1.98GB (28.0%)
2018-01-16T01:13:21.276+0000    [#######.................] loanDB.loans 616MB/1.98GB (30.4%)
2018-01-16T01:13:24.275+0000    [#######.................] loanDB.loans 662MB/1.98GB (32.7%)
2018-01-16T01:13:27.275+0000    [########................] loanDB.loans 711MB/1.98GB (35.1%)
2018-01-16T01:13:30.275+0000    [########................] loanDB.loans 760MB/1.98GB (37.5%)
2018-01-16T01:13:33.276+0000    [#########...............] loanDB.loans 807MB/1.98GB (39.8%)
2018-01-16T01:13:36.276+0000    [##########..............] loanDB.loans 855MB/1.98GB (42.2%)
2018-01-16T01:13:39.276+0000    [##########..............] loanDB.loans 904MB/1.98GB (44.6%)
2018-01-16T01:13:42.275+0000    [###########.............] loanDB.loans 951MB/1.98GB (46.9%)
2018-01-16T01:13:45.275+0000    [###########.............] loanDB.loans 962MB/1.98GB (47.5%)
2018-01-16T01:13:48.275+0000    [###########.............] loanDB.loans 999MB/1.98GB (49.3%)
2018-01-16T01:13:51.276+0000    [############............] loanDB.loans 1.02GB/1.98GB (51.7%)
2018-01-16T01:13:54.275+0000    [############............] loanDB.loans 1.07GB/1.98GB (54.1%)
2018-01-16T01:13:57.275+0000    [#############...........] loanDB.loans 1.12GB/1.98GB (56.5%)
2018-01-16T01:14:00.275+0000    [#############...........] loanDB.loans 1.16GB/1.98GB (58.8%)
2018-01-16T01:14:03.276+0000    [#############...........] loanDB.loans 1.21GB/1.98GB (61.3%)
```

Lessons Learned

1) Problem: Memory (not Storage) Issue on EC2

- t2.micro only comes with 1 GiB (~1 GB)
- Cannot successfully load data onto MongoDB (crashes at 25%)



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applicat
networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance t

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-
☐	General purpose	t2.nano	1	0.5	EBS only	
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	
☐	General purpose	t2.small	1	2	EBS only	
☐	General purpose	t2.medium	2	4	EBS only	

Lessons Learned

1) Solution: Increase Memory Size!

- With ~2G of data, t2.medium with 4 GiB Memory will successfully load the data into MongoDB

Currently selected: t2.medium (Variable ECUs, 2 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 4 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)
☐	General purpose	t2.nano	1	0.5	EBS only
☐	General purpose	t2.micro Free tier eligible	1	1	EBS only
☐	General purpose	t2.small	1	2	EBS only
☑	General purpose	t2.medium	2	4	EBS only
☐	General purpose	t2.large	2	8	EBS only
☐	General purpose	t2.xlarge	4	16	EBS only

MongoDB

```
> use loanDB
switched to db loanDB
> db.loans.aggregate({$sort: {"posted_time":1}})
assert: command failed: {
        "ok" : 0,
        "errmsg" : "Sort exceeded memory limit of 104857600 bytes, but did not op
t in to external sorting. Aborting operation. Pass allowDiskUse:true to opt in.",
        "code" : 16819,
        "codeName" : "Location16819"
} : aggregate failed
_getErrorWithCode@src/mongo/shell/utils.js:25:13
doassert@src/mongo/shell/assert.js:16:14
assert.commandWorked@src/mongo/shell/assert.js:403:5
DB.prototype._runAggregate@src/mongo/shell/db.js:260:9
DBCollection.prototype.aggregate@src/mongo/shell/collection.js:1212:12
@(shell):1:1

2018-01-17T15:05:20.655-0800 E QUERY    [thread1] Error: command failed: {
        "ok" : 0,
        "errmsg" : "Sort exceeded memory limit of 104857600 bytes, but did not op
t in to external sorting. Aborting operation. Pass allowDiskUse:true to opt in.",
        "code" : 16819,
        "codeName" : "Location16819"
} : aggregate failed :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
doassert@src/mongo/shell/assert.js:16:14
assert.commandWorked@src/mongo/shell/assert.js:403:5
DB.prototype._runAggregate@src/mongo/shell/db.js:260:9
DBCollection.prototype.aggregate@src/mongo/shell/collection.js:1212:12
@(shell):1:1
>
```

MongoDB

```
> db.loans.createIndex({"posted_time":1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.loans.aggregate({$sort: {"posted_time":1}})
{ "_id" : ObjectId("5a5d5af18259e57edf15e730"), "id" : 173, "name" : "", "origina
l_language" : "", "original_description" : "", "translated_description" : "", "fu
nded_amount" : 600, "loan_amount" : 600, "status" : "funded", "image_id" : "", "v
ideo_id" : "", "activity" : "Construction", "sector" : "Construction", "use" : ""
, "country_code" : "GZ", "country_name" : "Gaza", "town" : "", "currency_policy"
: "not shared", "currency_exchange_coverage_rate" : "", "currency" : "USD", "part
ner_id" : 2, "posted_time" : "2006-04-16 07:10:50+00:00", "planned_expiration_tim
e" : "", "disbursed_time" : "2005-04-14 05:27:55+00:00", "funded_time" : ISODate(
"1970-01-01T00:00:00Z"), "term_in_months" : 12, "lender_count" : 7, "journal_entr
ies_count" : 14, "bulk_journal_entries_count" : 7, "tags" : "user_favorite", "bor
rower_names" : "", "borrower_genders" : "", "borrower_pictured" : "", "repayment_
interval" : "bullet", "distribution_model" : "field_partner" }
```

DataFrame Creation on EC2

```
In [1]: from pyspark.sql import SparkSession

In [2]: my_spark = SparkSession.builder.appName("myApp").config("spark.mongodb.input.uri", "mongodb://127.0.0.1/loanDB.loans").c

In [3]: df = spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri","mongodb://127.0.0.1/loanDB.loans").load()

In [4]: df.show(2)
```

```
+--------------------+--------------------+---------------+--------------+----------------+--------------------
---+------------+------------+--------+-------------------------------+--------------+--------------------+---------
---------+-------------+--------------------+------+--------+---------------------+-----------+-----------+---------
------+--------------------+-----------------+----------+---------------------+--------------------+--------------
---+-------+------+----+-------------+--------+---------------------+--------------------+--------+
|                 _id|            activity|borrower_genders| borrower_names|borrower_pictured|bulk_journal_entries_co
unt|country_code|country_name|currency|currency_exchange_coverage_rate|currency_policy|      disbursed_time|distribut
ion_model|funded_amount|         funded_time|    id|image_id|journal_entries_count|lender_count|loan_amount|
  name|original_description|original_language|partner_id|planned_expiration_time|         posted_time|repayment_inter
val| sector|status|tags|term_in_months|    town|translated_description|                 use|video_id|
+--------------------+--------------------+---------------+--------------+----------------+--------------------
---+------------+------------+--------+-------------------------------+--------------+--------------------+---------
---------+-------------+--------------------+------+--------+---------------------+-----------+-----------+---------
------+--------------------+-----------------+----------+---------------------+--------------------+--------------
---+-------+------+----+-------------+--------+---------------------+--------------------+--------+
|[5a5d5aef8259e57e...|Personal Housing ...|         female|Norma Elizabeth|             true|
  3|          SV| El Salvador|     USD|                            0.1|         shared|2012-08-16 07:00:...|     fiel
d_partner|       1000.0|2012-09-27 13:43:...|466993| 1180943|                    5|          36|     1000.0|Norma Eli
zabeth|Norma tiene 38 añ...|          Spanish|       199|   2012-10-10 17:20:...|2012-08-28 21:49:...|           mont
hly|Housing|funded|     |           20|Anamoros|  Norma is 38 years...|to improve her ho...|        |
|[5a5d5aef8259e57e...|              Retail|         female|    Sayda Maria|             true|
  1|          NI|   Nicaragua|     USD|                                |     not shared|2012-08-23 07:00:...|     fiel
d_partner|        225.0|2012-09-13 10:01:...|468275| 1185373|                    2|           9|      225.0|    Sayda
Maria|sayda es originar...|          Spanish|       176|   2012-10-13 01:10:...|2012-08-31 20:29:...|           month
ly| Retail|funded|     |           11|   Boaco|  Sayda is original...|to buy rice, suga...|        |
+--------------------+--------------------+---------------+--------------+----------------+--------------------
```

Lessons Learned

2) Problem: Mixed Data Type

- ex. Column with Integer and String types.

```
In [35]: df2.groupBy(df2["term_in_months"]).count().orderBy("count",ascending=False).show()
```

```
/usr/local/Cellar/apache-spark/2.2.0/libexec/python/lib/py4j-0.10.4-src.zip/py4j/protocol.py in get_return_value(an
swer, gateway_client, target_id, name)
    317                 raise Py4JJavaError(
    318                     "An error occurred while calling {0}{1}{2}.\n".
--> 319                     format(target_id, ".", name), value)
    320             else:
    321                 raise Py4JError(

Py4JJavaError: An error occurred while calling o532.showString.
: org.apache.spark.SparkException: Job aborted due to stage failure: Task 6 in stage 63.0 failed 1 times, most rece
nt failure: Lost task 6.0 in stage 63.0 (TID 6009, localhost, executor driver): com.mongodb.spark.exceptions.MongoT
ypeConversionException: Cannot cast STRING into a IntegerType (value: BsonString{value=''})
        at com.mongodb.spark.sql.MapFunctions$.com$mongodb$spark$sql$MapFunctions$$convertToDataType(MapFunctions.s
cala:83)
        at com.mongodb.spark.sql.MapFunctions$$anonfun$3.apply(MapFunctions.scala:39)
        at com.mongodb.spark.sql.MapFunctions$$anonfun$3.apply(MapFunctions.scala:37)
        at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
        at scala.collection.TraversableLike$$anonfun$map$1.apply(TraversableLike.scala:234)
        at scala.collection.IndexedSeqOptimized$class.foreach(IndexedSeqOptimized.scala:33)
```

Lessons Learned

2) Solution: Preprocessing in Mongo

- ex. Remove rows/documents with empty strings.

```
> show dbs
admin    0.000GB
config   0.000GB
hw4      0.001GB
loanDB   1.548GB
local    0.000GB
msan697  0.000GB
> use loanDB
switched to db loanDB
> db.loans.remove({"term_in_months" : ""})
WriteResult({ "nRemoved" : 24 })
>
```

SparkSQL on EC2

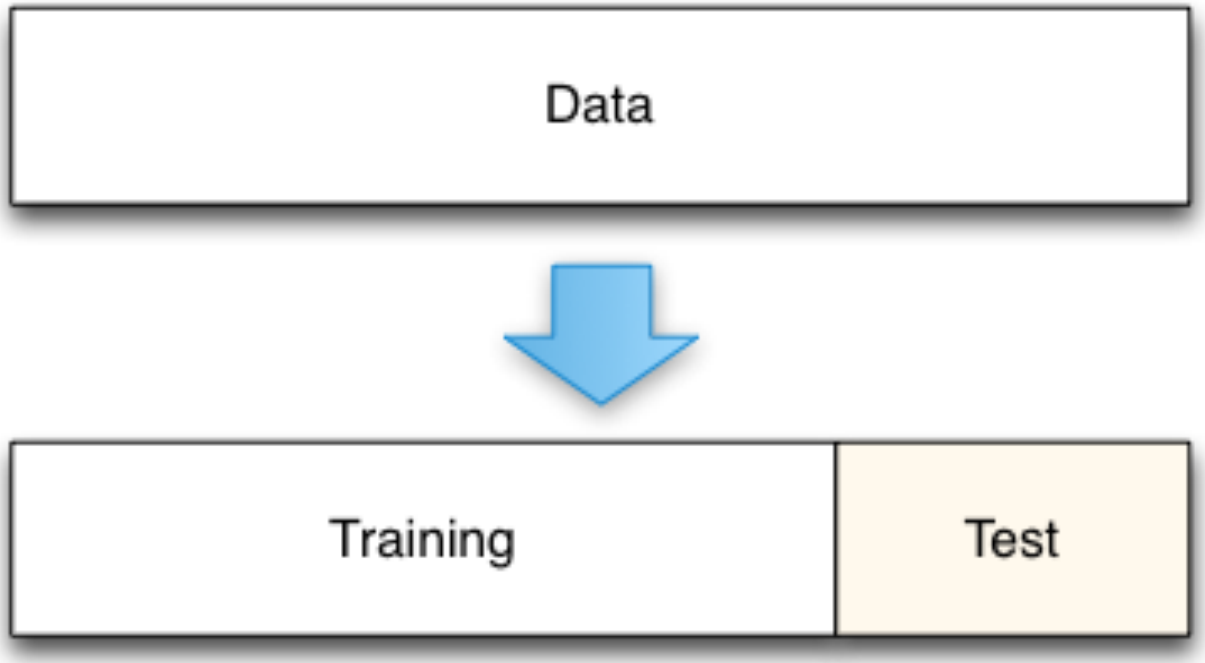
```
In [7]: df.groupBy(df['borrower_genders']).count().orderBy("count",ascending=False).show()
```

```
+--------------------+------+
|    borrower_genders| count|
+--------------------+------+
|              female|810367|
|                male|286190|
|                    | 29251|
|female, female, f...| 20151|
|      female, female| 18144|
|female, female, f...| 17333|
|female, female, f...| 16583|
|female, female, f...|  6112|
|female, female, f...|  5971|
|female, female, f...|  4295|
|female, female, f...|  4083|
|female, female, f...|  2834|
|female, female, f...|  2737|
|female, female, f...|  2493|
|female, female, f...|  1835|
|        male, female|  1786|
|        female, male|  1729|
|female, female, f...|  1679|
|female, female, f...|  1643|
|female, female, f...|  1579|
+--------------------+------+
only showing top 20 rows
```

Lessons Learned

3) Problem: Preprocessing data on Spark is not efficient

- ex. Splitting Data (without shuffle/randomization)
- ex. Changing schema types for Dates



In [5]:
```
df.printSchema()
```

```
root
 |-- _id: struct (nullable = true)
 |    |-- oid: string (nullable = true)
 |-- activity: string (nullable = true)
 |-- borrower_genders: string (nullable = true)
 |-- borrower_names: string (nullable = true)
 |-- borrower_pictured: string (nullable = true)
 |-- bulk_journal_entries_count: integer (nullable = true)
 |-- country_code: string (nullable = true)
 |-- country_name: string (nullable = true)
 |-- currency: string (nullable = true)
 |-- currency_exchange_coverage_rate: string (nullable = true)
 |-- currency_policy: string (nullable = true)
 |-- disbursed_time: string (nullable = true)
 |-- distribution_model: string (nullable = true)
 |-- funded_amount: double (nullable = true)
 |-- funded_time: string (nullable = true)
 |-- id: integer (nullable = true)
 |-- image_id: string (nullable = true)
 |-- journal_entries_count: integer (nullable = true)
 |-- lender_count: integer (nullable = true)
 |-- loan_amount: double (nullable = true)
 |-- name: string (nullable = true)
 |-- original_description: string (nullable = true)
 |-- original_language: string (nullable = true)
 |-- partner_id: string (nullable = true)
 |-- planned_expiration_time: string (nullable = true)
 |-- posted_time: string (nullable = true)
 |-- repayment_interval: string (nullable = true)
 |-- sector: string (nullable = true)
 |-- status: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- term_in_months: integer (nullable = true)
 |-- town: string (nullable = true)
 |-- translated_description: string (nullable = true)
 |-- use: string (nullable = true)
 |-- video_id: string (nullable = true)
```

Machine Learning (Spark ML) on EC2

In [16]:
```
#from pyspark.ml import Pipeline
from pyspark.ml.regression import RandomForestRegressor
#from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator
```

In [17]:
```
rf = RandomForestRegressor(featuresCol="features")
```

In [18]:
```
rfmodel = rf.fit(trainlpoints)
```

In [19]:
```
from pyspark.ml.evaluation import RegressionEvaluator
```

In [20]:
```
rfpredicts = rfmodel.transform(testlpoints)
evaluator = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(rfpredicts)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)
```

```
Root Mean Squared Error (RMSE) on test data = 103.361
```

Machine Learning (Spark ML) on EC2

In [21]:
```
eval2 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = eval2.evaluate(rfpredicts)
print("R-squared on test data = %g" % r2)
```

```
R-squared on test data = 0.835617
```

In [22]:
```
trainpredicts = rfmodel.transform(trainlpoints)
eval3 = RegressionEvaluator(
    labelCol="label", predictionCol="prediction", metricName="r2")
r2 = eval3.evaluate(trainpredicts)
print("R-squared on train data = %g" % r2)
```

```
R-squared on train data = 0.826539
```

Future Plans

- Rudimentary model yielded decent results
- More Feature Engineering:
  - NLP feature extraction
  - Add detailed date variables
- Random Forest Interpretation
- Move to more complex models:
  - Neural Networks

Thank you!

KIVA

Loans that change lives