In [ ]:
```
# getting data from the Google Cloud
!gsutil cp gs://yelp-dataset-bucket/* .
```

```
Copying gs://yelp-dataset-bucket/data-munging.ipynb...
Copying gs://yelp-dataset-bucket/getting-data.ipynb...
Copying gs://yelp-dataset-bucket/map-reduce.ipynb...
Copying gs://yelp-dataset-bucket/mapper.py...
/ [4 files][ 26.6 KiB/ 26.6 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying gs://yelp-dataset-bucket/reducer.py...
Copying gs://yelp-dataset-bucket/spark-ml.ipynb...
Copying gs://yelp-dataset-bucket/working-with-hdfs.ipynb...
Copying gs://yelp-dataset-bucket/yelp_academic_dataset_review.json...
/ [8 files][  5.9 GiB/  5.9 GiB]   51.7 MiB/s
Operation completed over 8 objects/5.9 GiB.
```

In [ ]:
```
# you can download data from Kaggle directly using their package
# https://github.com/Kaggle/kaggle-api
!pip install kaggle
```

```
Collecting kaggle
  Downloading https://files.pythonhosted.org/packages/99/33/365c0d13f07a2a54744d027fe20b60dacdfdfb33bc04746db6ad0b79340b/kaggle-1.5.10.tar.gz (59kB)
    100% |████████████████████████████████| 61kB 6.1MB/s eta 0:00:01
Requirement already satisfied: six>=1.10 in /opt/conda/anaconda/lib/python2.7/site-packages (from kaggle)
Requirement already satisfied: certifi in /opt/conda/anaconda/lib/python2.7/site-packages (from kaggle)
Requirement already satisfied: python-dateutil in /opt/conda/anaconda/lib/python2.7/site-packages (from kaggle)
Requirement already satisfied: requests in /opt/conda/anaconda/lib/python2.7/site-packages (from kaggle)
Collecting tqdm (from kaggle)
  Downloading https://files.pythonhosted.org/packages/8a/54/115f0c28a61d56674c3a5e05c46d6c3523ad196e1dcd3e2d8b119026df36/tqdm-4.54.1-py2.py3-none-any.whl (69kB)
    100% |████████████████████████████████| 71kB 5.6MB/s eta 0:00:01
Collecting python-slugify (from kaggle)
  Downloading https://files.pythonhosted.org/packages/9f/42/e336f96a8b6007428df772d0d159b8eee9b2f1811593a4931150660402c0/python-slugify-4.0.1.tar.gz
Requirement already satisfied: urllib3 in /opt/conda/anaconda/lib/python2.7/site-packages (from kaggle)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/anaconda/lib/python2.7/site-packages (from requests->kaggle)
Requirement already satisfied: idna<2.7,>=2.5 in /opt/conda/anaconda/lib/python2.7/site-packages (from requests->kaggle)
```

```
Collecting text-unidecode>=1.3 (from python-slugify->kaggle)
  Downloading https://files.pythonhosted.org/packages/a6/a5/c0b6468d3824fe3fde30dbb5e1f687b291608f9473681bbf7dabbf5a87d
7/text_unidecode-1.3-py2.py3-none-any.whl (78kB)
    100% |████████████████████████████████| 81kB 7.0MB/s eta 0:00:01
Building wheels for collected packages: kaggle, python-slugify
  Running setup.py bdist_wheel for kaggle ... done
  Stored in directory: /root/.cache/pip/wheels/3a/d1/7e/6ce09b72b770149802c653a02783821629146983ee5a360f10
  Running setup.py bdist_wheel for python-slugify ... done
  Stored in directory: /root/.cache/pip/wheels/67/b8/ba/041548f30a6fc058c9b3f79a5b7b6aea925a15dd1e5c4992a4
Successfully built kaggle python-slugify
Installing collected packages: tqdm, text-unidecode, python-slugify, kaggle
Successfully installed kaggle-1.5.10 python-slugify-4.0.1 text-unidecode-1.3 tqdm-4.54.1
You are using pip version 9.0.1, however version 20.3.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
```

In [ ]:

```
!zip -9 some.zip /home/borisshminke/some.model/*
```

```
  adding: home/borisshminke/some.model/metadata/ (stored 0%)
  adding: home/borisshminke/some.model/stages/ (stored 0%)
```

In [ ]:

```
# if you have a zipped data, you should unzip it before uploading to HDFS
# storing zip-files on Googl Cloud is better
!unzip some.zip
```

```
Archive:  some.zip
```

In [ ]:

```
# uploading your data to HDFS
!hdfs dfs -put yelp_academic_dataset_review.json /user/borisshminke
```

In [ ]:

```
# reading your data, JSON and CSV are preferred if using Spark
data = (
    spark.read
    .json("/user/borisshminke/yelp_academic_dataset_review.json")
)
```

In [ ]:

```
# feature engineering
from pyspark.ml.pipeline import Pipeline
from pyspark.ml.feature import Tokenizer, HashingTF, IDF
from pyspark.ml.regression import LinearRegression
```

```
pipeline = Pipeline(stages=[
    Tokenizer(inputCol="text", outputCol="words"),
    HashingTF(inputCol="words", outputCol="term_frequency"),
    IDF(inputCol="term_frequency", outputCol="features"),
    LinearRegression(labelCol="stars")
])
```

In [ ]:

```
# your param grid, use at least two options
from pyspark.ml.tuning import ParamGridBuilder

param_grid = (
    ParamGridBuilder()
    .addGrid("regParam", [0])
    .build()
)
```

In [ ]:

```
# use a small fraction of data for debug
# if running on all the data lasts forever, you can create a larger cluster
# or if you run out of credits, don't wory, send an working copy on sample
debug_data = data.sample(0.01).cache()
```

In [ ]:

```
# you can use cross validation here, or split on train and test manually
from pyspark.ml.tuning import TrainValidationSplit
from pyspark.ml.evaluation import RegressionEvaluator

models = TrainValidationSplit(
    estimator=pipeline,
    estimatorParamMaps=param_grid,
    evaluator=RegressionEvaluator(labelCol="stars")
).fit(debug_data)
```

In [ ]:

```
# for a classification use a tangible metric
# http://spark.apache.org/docs/2.4.3/api/python/pyspark.ml.html#module-pyspark.ml.evaluation
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

MulticlassClassificationEvaluator(metricName="accuracy")
```

Out[ ]: MulticlassClassificationEvaluator_42a29d5ede0a3a45f6b4

In [ ]:
```
# reporting values for training set is not necessary
models.validationMetrics
```

Out[ ]:
```
[2.27781356143004]
```

In [ ]:
```
# be sure to use the right metric:)
models.getEvaluator().getMetricName()
```

Out[ ]:
```
'rmse'
```

In [ ]:
```
# the deadline is Jan 8th
```

In [ ]:
```
# fitting model on all data (without splits)
some_model = pipeline.fit(debug_data)
```

In [ ]:
```
some_model
```

Out[ ]:
```
PipelineModel_4bcf8c3f854bcd357c1b
```

In [ ]:
```
# save a train model
some_model.write().overwrite().save("/user/borisshminke/some.model")
```

In [ ]:
```
# check that the model was saved
!hdfs dfs -ls /user/borisshminke/some.model/stages
```

```
Found 4 items
drwxr-xr-x   - root hadoop          0 2020-12-17 14:05 /user/borisshminke/some.model/stages/0_Tokenizer_4e15be7484ded4f
2fdff
drwxr-xr-x   - root hadoop          0 2020-12-17 14:05 /user/borisshminke/some.model/stages/1_HashingTF_4a109dbe8d4c890
f79a0
drwxr-xr-x   - root hadoop          0 2020-12-17 14:05 /user/borisshminke/some.model/stages/2_IDF_438581dbe3d72dc01450
drwxr-xr-x   - root hadoop          0 2020-12-17 14:05 /user/borisshminke/some.model/stages/3_LinearRegression_4aa58c21
69880984a486
```

In [ ]:
```
# get the model from HDFS
!hdfs dfs -get /user/borisshminke/some.model /home/borisshminke/some.model
```

In [ ]:
```
# uploading your model to Google Cloud Storage
!gsutil cp -r /home/borisshminke/some.model gs://yelp-dataset-bucket/
```

```
Copying file:///home/borisshminke/some.model/stages/2_IDF_438581dbe3d72dc01450/metadata/part-00000 [Content-Type=applic
ation/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/2_IDF_438581dbe3d72dc01450/metadata/_SUCCESS [Content-Type=applicat
ion/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/2_IDF_438581dbe3d72dc01450/data/part-00000-848624a9-7100-4804-8e2c-
44ccdd011e34-c000.snappy.parquet [Content-Type=application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/2_IDF_438581dbe3d72dc01450/data/_SUCCESS [Content-Type=application/
octet-stream]...
/ [4 files][248.4 KiB/248.4 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying file:///home/borisshminke/some.model/stages/3_LinearRegression_4aa58c2169880984a486/metadata/part-00000 [Conten
t-Type=application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/3_LinearRegression_4aa58c2169880984a486/metadata/_SUCCESS [Content-
Type=application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/3_LinearRegression_4aa58c2169880984a486/data/part-00000-efa39c68-0c
ca-4f23-bd92-1be7877cdfbf-c000.snappy.parquet [Content-Type=application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/3_LinearRegression_4aa58c2169880984a486/data/_SUCCESS [Content-Type
=application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/1_HashingTF_4a109dbe8d4c890f79a0/metadata/part-00000 [Content-Type=
application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/1_HashingTF_4a109dbe8d4c890f79a0/metadata/_SUCCESS [Content-Type=ap
plication/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/0_Tokenizer_4e15be7484ded4f2fdff/metadata/part-00000 [Content-Type=
application/octet-stream]...
Copying file:///home/borisshminke/some.model/stages/0_Tokenizer_4e15be7484ded4f2fdff/metadata/_SUCCESS [Content-Type=ap
plication/octet-stream]...
Copying file:///home/borisshminke/some.model/metadata/part-00000 [Content-Type=application/octet-stream]...
Copying file:///home/borisshminke/some.model/metadata/_SUCCESS [Content-Type=application/octet-stream]...
\ [14 files][  1.9 MiB/  1.9 MiB]
Operation completed over 14 objects/1.9 MiB.
```

In [ ]:
```
# you can load the model which you saved previously
from pyspark.ml.pipeline import PipelineModel
```

```
some_model = PipelineModel.read().load("/user/borisshminke/some.model")
```

In [ ]:
```
some_predictions = some_model.transform(debug_data)
```

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

RegressionEvaluator(labelCol="stars").evaluate(
    some_predictions
)
```

Out[ ]:
```
2.3824686010483127
```

In [ ]:
```
debug_data.columns
```

Out[ ]:
```
['business_id',
 'cool',
 'date',
 'funny',
 'review_id',
 'stars',
 'text',
 'useful',
 'user_id']
```

In [ ]:
```
print(debug_data.count())
print(debug_data.dropna().count())
```

```
79613
79613
```

In [ ]:
```
# how to do scaling and prepare number columns for feeding a model
from pyspark.ml.feature import VectorAssembler, MinMaxScaler

pipeline = Pipeline(stages=[
    VectorAssembler(
        inputCols=["funny", "useful", "cool"],
        outputCol="pre_features"
    ),
```

```
    MinMaxScaler(inputCol="pre_features", outputCol="features")
])
```

In [ ]:

```
(
    pipeline.fit(debug_data).transform(debug_data)
    .select("funny", "useful", "cool", "features", "pre_features")
).show()
```

```
+-----+------+----+--------------------+-------------+
|funny|useful|cool|            features| pre_features|
+-----+------+----+--------------------+-------------+
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    1|     6|   1|[0.00584795321637...|[1.0,6.0,1.0]|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    1|     2|   0|[0.00584795321637...|[1.0,2.0,0.0]|
|    0|     1|   0|[0.0,0.0067114093...|[0.0,1.0,0.0]|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    0|     1|   0|[0.0,0.0067114093...|[0.0,1.0,0.0]|
|    0|     1|   0|[0.0,0.0067114093...|[0.0,1.0,0.0]|
|    0|     0|   1|[0.0,0.0,0.007194...|[0.0,0.0,1.0]|
|    0|     0|   1|[0.0,0.0,0.007194...|[0.0,0.0,1.0]|
|    0|     5|   0|[0.0,0.0335570469...|[0.0,5.0,0.0]|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    1|     1|   5|[0.00584795321637...|[1.0,1.0,5.0]|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    1|     5|   4|[0.00584795321637...|[1.0,5.0,4.0]|
|    0|     0|   0|       [0.0,0.0,0.0]|    (3,[],[])|
|    0|     1|   1|[0.0,0.0067114093...|[0.0,1.0,1.0]|
|    0|     2|   0|[0.0,0.0134228187...|[0.0,2.0,0.0]|
+-----+------+----+--------------------+-------------+
only showing top 20 rows
```

In [ ]:

```
debug_data.summary().toPandas()
```

Out[ ]:

	summary	business_id	cool	date	funny	review_id	stars	text
0	count	79613	79613	79613	79613	79613	79613	79613
1	mean	None	0.5793651790536721	None	0.46796377476040346	None	3.706228882217728	None
2	stddev	None	2.503847667428883	None	2.1866375974104084	None	1.489918562112281	None

	summary	business_id	cool	date	funny	review_id	stars	text
3	min	-1UhMGODdWsrMastO9DZw	0	2005-03-16 17:08:51	0	--0pfY3vQilgl20btE0fVQ	1.0	! ! ! BEST MASSAGE THERAPIST IN TOWN ! ! ! \nA...
4	25%	None	0	None	0	None	3.0	None
5	50%	None	0	None	0	None	4.0	None
6	75%	None	0	None	0	None	5.0	None
7	max	zzwicjPC9g246MK2M1ZFBA	139	2019-12-13 15:22:44	171	zzsSYtKmFzbg5as5n4LS_Q	5.0	（忘记照相了，也忘记菜名了...所以盗用了一些大家的图片）点了下面图上这几样小菜和面，味道感...

In [ ]:
```
debug_data.select("funny").rdd.take(10)
```

Out[ ]:
```
[Row(funny=0),
 Row(funny=1),
 Row(funny=0),
 Row(funny=1),
 Row(funny=0),
 Row(funny=0),
 Row(funny=0),
 Row(funny=0),
 Row(funny=0),
 Row(funny=0)]
```

In [ ]:
```
# histograms are not ported yet to DataFrames API
# so you need to use RDDs
(
    debug_data.select("funny")
    .rdd.map(lambda row: row[0])
```

```
    .histogram(10)
)
```

Out[ ]:
```
([0.0,
  17.1,
  34.2,
  51.300000000000004,
  68.4,
  85.5,
  102.60000000000001,
  119.70000000000002,
  136.8,
  153.9,
  171],
 [79442, 133, 14, 12, 7, 3, 0, 0, 0, 2])
```

In [ ]:
```
debug_data.select("business_id").distinct().count()
```

Out[ ]:
```
41062
```

In [ ]:
```
# for categorical variables you can do one-hot encoding
from pyspark.ml.feature import OneHotEncoder, StringIndexer

pipeline = Pipeline(stages=[
    StringIndexer(inputCol="business_id", outputCol="category_id"),
    OneHotEncoder(inputCol="category_id", outputCol="one"),
    VectorAssembler(
        inputCols=["funny", "useful", "cool", "one"],
        outputCol="pre_features"
    ),

])
```

In [ ]:
```
(
    pipeline.fit(debug_data).transform(debug_data)
    .select("pre_features")
    .limit(10)
).toPandas()
```

Out[ ]:

pre_features

	pre_features
0	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1	(1.0, 6.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3	(1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
6	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
7	(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
8	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
9	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...

In [ ]: