Paul 3 - evaluate C...

```
%pyspark                                                                        FINISHED

# Zeppelin notebook to evaluate CC domain feature vectors against the DNS-BH Malware
# Domain Blocklist from this site: http://mirror1.malwaredomains.com
# Specifically, the 'justdomains' file, which currently contains 31k 'bad' domains.
# We train using some of these domains, and try to predict the rest from amongst all
# the domains for which we have feature vectors.
# PJ - 21 Sept 2017

import boto
from pyspark.sql.types import *

# Import the DNS-BH domain list as a DataFrame
bh_schema=StructType([StructField("domain", StringType(), False)])
dns_bh=spark.read.csv('s3://billsdata.net/CommonCrawl/DNS-BH/justdomains.dms', header=F
dns_bh.show(3)
print("Bad domains in DNS-BH: " + str(dns_bh.count()))
dns_bh.cache()
```

```
+--------------------+
|              domain|
+--------------------+
|amazon.co.uk.secu...|
|autosegurancabras...|
|christianmensfell...|
+--------------------+
only showing top 3 rows
Bad domains in DNS-BH: 31877
DataFrame[domain: string]
```

```
%pyspark                                                                        FINISHED

# Load Bill's domain feature vectors from s3, in the following format:
# (u'www.angelinajolin.com', [4.30406509320417, 0.02702702702702703, 0.0, 0.13513513513

nfiles=128 # (takes about 5 mins for 128 files)

# Load feature vectors from WAT files (from 'Bill 6' notebook) as an RDD:
inputURI = "s3://billsdata.net/CommonCrawl/domain_hex_feature_vectors_from_%d_WAT_files
features_rdd = sc.textFile(inputURI).map(eval)
import pyspark.sql.types as typ
schema=StructType([StructField("domain", StringType(), False), StructField("vector", Ar
features_df=spark.createDataFrame(features_rdd,schema)
features_df.cache()
print("Nr domains:", features_df.count())
print(features_df.show(1))
features_df.printSchema()
```

```
('Nr domains:', 2626203)
+-----------+--------------------+
|     domain|              vector|
+-----------+--------------------+
|www.iggl.de|[3.63758615972638...|
+-----------+--------------------+
only showing top 1 row
None
root
 |-- domain: string (nullable = false)
 |-- vector: array (nullable = true)
 |    |-- element: double (containsNull = false)
```

```
%pyspark                                                                          FINISHED

# Spark.ML classifiers require VectorUDF type, rather than Array, so we need to convert
from pyspark.ml.linalg import Vectors, VectorUDT
from pyspark.sql.functions import UserDefinedFunction
vectorize=UserDefinedFunction(lambda vs: Vectors.dense(vs), VectorUDT())
features_df = features_df.withColumn("vec", vectorize(features_df['vector'])).drop('vec
features_df.show(1)
features_df.printSchema()
```

```
+-----------+--------------------+
|     domain|                 vec|
+-----------+--------------------+
|www.iggl.de|[3.63758615972638...|
+-----------+--------------------+
only showing top 1 row
root
 |-- domain: string (nullable = false)
 |-- vec: vector (nullable = true)
```

```
%pyspark                                                                          FINISHED

# Remove www. prefix
from pyspark.sql.functions import UserDefinedFunction
from pyspark.sql.types import StringType

name='domain'
prefix="www."
udf = UserDefinedFunction(lambda x: (x[len(prefix):] if (x.startswith(prefix) if x else

dns_bh2 = dns_bh.select(*[udf(column).alias(name) if column == name else column for col
dns_bh2.show(3)
```

```
+--------------------+
|              domain|
+--------------------+
|amazon.co.uk.secu...|
|autosegurancabras...|
|christianmensfell...|
+--------------------+
only showing top 3 rows
```

```
%pyspark                                                                    FINISHED

# Filter feature vectors for only those vectors that have entries in the DNS-BH list di
common_domains_df=features_df.join(dns_bh2, ["domain"]) # doesn't create extra column
common_domains_df.cache()
features_df.unpersist()
dns_bh.unpersist()
print("Number of labelled domains = " + str(common_domains_df.count()))
common_domains_df.show(3)
common_domains_df.printSchema()

# Damn, we appear to only have 57 bad domains in the 128 WAT files currently processed |
# TODO: Get more/better data, both for CC feature vectors, and known bad domains!
```

```
Number of labelled domains = 57
+--------------------+--------------------+
|              domain|              vector|
+--------------------+--------------------+
|         tsjyoti.com|[4.57471097850338...|
|     jur-science.com|[6.52356230614951...|
|fernandovillamorj...|[4.26267987704131...|
+--------------------+--------------------+
only showing top 3 rows
root
 |-- domain: string (nullable = false)
 |-- vector: array (nullable = true)
 |    |-- element: double (containsNull = false)
```

```
%pyspark                                                                      ERROR

# Create numeric indexes for our classes
from pyspark.ml.feature import IndexToString, StringIndexer, VectorIndexer
labelIndexer = StringIndexer(inputCol="category", outputCol="indexedCategory").fit(comm

# Split into training and test sets using spark.ML API
domains_train, domains_test = common_domains_df.randomSplit([0.7,0.3],seed=42)

# TODO: Need to add a Good class, and bring in domains from Bill's data
#       Then add 'Good' and 'Bad' labels to each record for training and test.

#domains_test.groupBy('category').count().show()
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 355, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 2, in <module>
  File "/usr/lib/spark/python/pyspark/ml/base.py", line 64, in fit
    return self._fit(dataset)
  File "/usr/lib/spark/python/pyspark/ml/wrapper.py", line 265, in _fit
    java_model = self._fit_java(dataset)
  File "/usr/lib/spark/python/pyspark/ml/wrapper.py", line 262, in _fit_java
    return self._java_obj.fit(dataset._jdf)
  File "/usr/lib/spark/python/lib/py4j-0.10.4-src.zip/py4j/java_gateway.py", line 1133,
in __call__
    answer, self.gateway_client, self.target_id, self.name)
  File "/usr/lib/spark/python/pyspark/sql/utils.py", line 79, in deco
    raise IllegalArgumentException(s.split(': ', 1)[1], stackTrace)
```

```
%pyspark                                                                    ERROR

# Create a pipeline and fit a RandomForest Classifier using spark.ml
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier

# Build our RF classifier
rf = RandomForestClassifier(labelCol="indexedCategory", featuresCol="vec", numTrees=10)

# Convert indexed labels back to original labels
labelConverter = IndexToString(inputCol="prediction", outputCol="predictedCategory", lal

# Define and run the full pipeline to train the model and make predictions
pipeline = Pipeline(stages=[labelIndexer, rf, labelConverter])
model=pipeline.fit(domains_train)
predictions=model.transform(domains_test)
print(predictions.take(1))
predictions.select("predictedCategory", "category", "vec").show(5)
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 355, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 4, in <module>
NameError: name 'labelIndexer' is not defined
```

```
%pyspark                                                                    ERROR

# Select (prediction, true label) and compute test error
```

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator1 = BinaryClassificationEvaluator(labelCol="indexedCategory", predictionCol="p
evaluator2 = BinaryClassificationEvaluator(labelCol="indexedCategory", predictionCol="p

accuracy = evaluator1.evaluate(predictions)
f1=evaluator2.evaluate(predictions)

print("Accuracy=%g, F1=%g" % (accuracy, f1))
```

```
Exception AttributeError: "'BinaryClassificationEvaluator' object has no attribute '_jav
a_obj'" in <object repr() failed> ignored

Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-1849060203705397415.py", line 355, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 2, in <module>
  File "/usr/lib/spark/python/pyspark/__init__.py", line 104, in wrapper
    return func(self, **kwargs)
TypeError: __init__() got an unexpected keyword argument 'predictionCol'
```

```
%pyspark
```

READY