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

import matplotlib.pyplot as plt
%matplotlib inline

from pyspark.sql import SparkSession
from pyspark.ml import Pipeline
from pyspark.ml.feature import CountVectorizer,StringIndexer, RegexTokenizer,Sto
from pyspark.sql.functions import col, udf,regexp_replace,isnull
from pyspark.sql.types import StringType,IntegerType
from pyspark.ml.classification import NaiveBayes, RandomForestClassifier, Logist
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClass
```

In [ ]:

In [2]:
```
# Spark NLP
spark = SparkSession.builder \
    .appName("Spark NLP")\
    .master("local[4]")\
    .config("spark.driver.memory","16G")\
    .config("spark.driver.maxResultSize", "0") \
    .config("spark.kryoserializer.buffer.max", "2000M")\
    .config("spark.jars.packages", "com.johnsnowlabs.nlp:spark-nlp_2.12:3.3.2")\
    .getOrCreate()
```

In [3]:
```
#create Spark session
spark = SparkSession.builder.appName('RedditComments').getOrCreate()

#change configuration settings on Spark
/gateway/default/node/conf?host&port = spark.sparkContext._conf.setAll([('spark.

#print spark configuration settings
spark.sparkContext.getConf().getAll()
```

Out[3]:
```
[('spark.eventLog.enabled', 'true'),
 ('spark.dynamicAllocation.minExecutors', '1'),
 ('spark.sql.warehouse.dir', 'file:/spark-warehouse'),
 ('spark.history.fs.logDirectory',
  'gs://dataproc-temp-us-central1-84427460872-fixxspuh/97cd0fe0-90c7-4b68-ba8a-f
cc718886ab3/spark-job-history'),
 ('spark.executor.memory', '5g'),
 ('spark.driver.host',
  'cluster-e4d0-m.us-central1-b.c.big-data-platforms-329618.internal'),
 ('spark.yarn.am.memory', '640m'),
 ('spark.cores.max', '4'),
 ('spark.executor.cores', '4'),
 ('spark.app.startTime', '1638388665963'),
 ('spark.executor.instances', '2'),
 ('spark.driver.memory', '8g'),
 ('spark.serializer.objectStreamReset', '100'),
 ('spark.yarn.unmanagedAM.enabled', 'true'),
 ('spark.sql.autoBroadcastJoinThreshold', '43m'),
 ('spark.submit.deployMode', 'client'),
 ('spark.driver.port', '37961'),
```

```
 ('spark.ui.filters',
  'org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter'),
 ('spark.driver.appUIAddress',
  'http://cluster-e4d0-m.us-central1-b.c.big-data-platforms-329618.internal:3503
3'),
 ('spark.sql.cbo.joinReorder.enabled', 'true'),
 ('spark.driver.maxResultSize', '1920m'),
 ('spark.shuffle.service.enabled', 'true'),
 ('spark.org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter.param.PROXY_
URI_BASES',
  'http://cluster-e4d0-m:8088/proxy/application_1638310962676_0011'),
 ('spark.scheduler.mode', 'FAIR'),
 ('spark.org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter.param.PROXY_
HOSTS',
  'cluster-e4d0-m'),
 ('spark.sql.adaptive.enabled', 'true'),
 ('spark.yarn.jars', 'local:/usr/lib/spark/jars/*'),
 ('spark.app.id', 'application_1638310962676_0011'),
 ('spark.scheduler.minRegisteredResourcesRatio', '0.0'),
 ('spark.executor.id', 'driver'),
 ('spark.hadoop.hive.execution.engine', 'mr'),
 ('spark.hadoop.mapreduce.fileoutputcommitter.algorithm.version', '2'),
 ('spark.dynamicAllocation.maxExecutors', '10000'),
 ('spark.master', 'yarn'),
 ('spark.ui.port', '0'),
 ('spark.executorEnv.PYTHONPATH',
  '/usr/lib/spark/python/lib/py4j-0.10.9-src.zip:/usr/lib/spark/python/:<CPS>{{P
WD}}/pyspark.zip<CPS>{{PWD}}/py4j-0.10.9-src.zip'),
 ('spark.app.name', 'Spark Updated Conf'),
 ('spark.sql.catalogImplementation', 'hive'),
 ('spark.eventLog.dir',
  'gs://dataproc-temp-us-central1-84427460872-fixxspuh/97cd0fe0-90c7-4b68-ba8a-f
cc718886ab3/spark-job-history'),
 ('spark.rpc.message.maxSize', '512'),
 ('spark.rdd.compress', 'True'),
 ('spark.ui.proxyBase', '/proxy/application_1638310962676_0011'),
 ('spark.yarn.historyServer.address', 'cluster-e4d0-m:18080'),
 ('spark.submit.pyFiles', ''),
 ('spark.dynamicAllocation.enabled', 'true'),
 ('spark.history.fs.gs.outputstream.type', 'BASIC'),
 ('spark.yarn.isPython', 'true'),
 ('spark.executorEnv.OPENBLAS_NUM_THREADS', '1'),
 ('spark.ui.showConsoleProgress', 'true'),
 ('spark.sql.cbo.enabled', 'true')]
```

In [4]:

```
df = spark.read \
    .option("delimiter",",") \
    .option("multiLine","true") \
    .option("quote", "\"")  \
    .option("escape", "\"") \
    .option("ignoreLeadingWhiteSpace",True) \
    .csv("gs://reddit-data-team-1/data_cleaned.csv",inferSchema=True, header=Tru
```

```
21/12/01 19:57:55 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread
(Thread[GetFileInfo #1,5,main]) interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.j
ava:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(Flue
ntFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfte
rExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecu
```

```
te(HadoopThreadPoolExecutor.java:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.
java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecuto
r.java:624)
        at java.lang.Thread.run(Thread.java:748)
```

In [5]:

```
# df = spark.read \
#     .option("quote", "\"")  \
#     .option("escape", "\"") \
#     .option("ignoreLeadingWhiteSpace",True) \
#     .csv("gs://reddit-data-team-1/data_cleaned.csv",inferSchema=True, header=T
```

In [6]:

```
df.printSchema()
```

```
root
 |-- body: string (nullable = true)
 |-- clean_comment: string (nullable = true)
 |-- category: integer (nullable = true)
```

In [7]:

```
display(df)
```

```
DataFrame[body: string, clean_comment: string, category: int]
```

In [8]:

```
df.summary().show()
```

```
+-------+--------------------+--------------------+--------------------+
|summary|                body|       clean_comment|            category|
+-------+--------------------+--------------------+--------------------+
|  count|             4864687|             4849322|             4864688|
|   mean|            Infinity|                NaN|0.18691332311548037|
| stddev|                 NaN|                NaN| 0.7967911112011219|
|    min|       "Mother's Little...|    now playing
mumf...|                  -1|
|    25%|              3820.0|              1999.0|                   0|
|    50%|              6790.0|              5670.0|                   0|
|    75%|              9837.0|              9164.0|                   1|
|    max|                    |?? ?????????????...|                   1|
+-------+--------------------+--------------------+--------------------+
```

In [9]:

```
df.dtypes
```

Out[9]:

```
[('body', 'string'), ('clean_comment', 'string'), ('category', 'int')]
```

Preprocessing

In [10]:

```
data = df
```

In [11]:

```
from pyspark.sql.functions import col
```

```
data.groupBy("category").count().orderBy(col("count").desc()).show()
```

```
[Stage 5:>                                                          (0 + 1) / 1]
+--------+-------+
|category|  count|
+--------+-------+
|       1|2083852|
|       0|1606259|
|      -1|1174577|
+--------+-------+
```

In [12]:

```
#data = data.filter((col("category") == "1") | (col("category") == "0") | (col("
```

In [13]:

```
from pyspark.sql.functions import col

data.groupBy("category").count().orderBy(col("count").desc()).show()
```

```
[Stage 8:>                                                          (0 + 1) / 1]
+--------+-------+
|category|  count|
+--------+-------+
|       1|2083852|
|       0|1606259|
|      -1|1174577|
+--------+-------+
```

In [14]:

```
data = data.withColumn("clean_comment",regexp_replace(col('body'), '\d+', ''))
data.show(5)
```

```
+--------------------+--------------------+--------+
|                body|       clean_comment|category|
+--------------------+--------------------+--------+
|gg this one's ove...|gg this one's ove...|       0|
|No one has a Euro...|No one has a Euro...|       0|
|That the kid "..r...|That the kid "..r...|      -1|
|                NSFL|                NSFL|       0|
|Get back to your ...|Get back to your ...|       0|
+--------------------+--------------------+--------+
only showing top 5 rows
```

In [15]:

```
from pyspark.sql.functions import regexp_replace, trim, col, lower
def removePunctuation(column):
    """Removes punctuation, changes to lower case, and strips leading and traili

    Note:
        Only spaces, letters, and numbers should be retained.  Other characters
        eliminated (e.g. it's becomes its).  Leading and trailing spaces should
        punctuation is removed.

    Args:
        column (Column): A Column containing a sentence.

    Returns:
```

```
        Column: A Column named 'sentence' with clean-up operations applied.
    """
    return trim(lower(regexp_replace(column, '[^\sa-zA-Z0-9]', '''))).alias('sent

sentenceDF = sqlContext.createDataFrame([('Hi, you!',),
                                         (' No under_score!',),
                                         (' *      Remove punctuation then space
sentenceDF.show(truncate=False)
(sentenceDF
 .select(removePunctuation(col('sentence')))
 .show(truncate=False))
```

```
+-----------------------------------------+
|sentence                                 |
+-----------------------------------------+
|Hi, you!                                 |
| No under_score!                         |
| *      Remove punctuation then spaces  * |
+-----------------------------------------+

+------------------------------+
|sentence                      |
+------------------------------+
|hi you                        |
|no underscore                 |
|remove punctuation then spaces|
+------------------------------+
```

In [16]:

```
data = data.withColumn("cleaned",removePunctuation(col('clean_comment')))
```

In [17]:

```
data.show(2)
```

```
+--------------------+--------------------+--------+--------------------+
|                body|       clean_comment|category|             cleaned|
+--------------------+--------------------+--------+--------------------+
|gg this one's ove...|gg this one's ove...|       0|gg this ones over...|
|No one has a Euro...|No one has a Euro...|       0|no one has a euro...|
+--------------------+--------------------+--------+--------------------+
only showing top 2 rows
```

In [18]:

```
data.count()
```

Out[18]: 4864688

In [19]:

```
data = data.dropna()
data.count()
```

Out[19]: 4864687

Model Pipeline

In [20]:
```
from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.classification import LogisticRegression

# regular expression tokenizer
regexTokenizer = RegexTokenizer(inputCol="cleaned", outputCol="words", pattern="

# stop words
add_stopwords = ["http","https","amp","rt","t","c","the"]
#add_stopwords = ["http","https","amp","rt","t","c","the",'narendra','modi','...

stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setS

# bag of words count
#countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocab
countVectors = CountVectorizer(inputCol="filtered", outputCol="features")

label_stringIdx = StringIndexer(inputCol = "category", outputCol = "label")
```

In [21]:
```
from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler

pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors, labe
```

In [22]:
```
pipelineFit = pipeline.fit(data)
```

In [23]:
```
#pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors])

# Fit the pipeline to training documents.
pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
dataset.show(5)
```

```
21/12/01 20:15:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
[Stage 37:>                                                         (0 + 1) / 1]
+--------------------+--------------------+--------+--------------------+-------
-------------+--------------------+--------------------+-----+
|                body|       clean_comment|category|             cleaned|
words|            filtered|            features|label|
+--------------------+--------------------+--------+--------------------+-------
-------------+--------------------+--------------------+-----+
|gg this one's ove...|gg this one's ove...|       0|gg this ones over...|[gg, th
is, ones, ...|[gg, this, ones, ...|(262144,[0,2,14,1...|  1.0|
|No one has a Euro...|No one has a Euro...|       0|no one has a euro...|[no, on
e, has, a,...|[no, one, has, a,...|(262144,[1,7,12,1...|  1.0|
|That the kid "..r...|That the kid "..r...|      -1|that the kid remi...|[that,
the, kid, ...|[that, kid, remin...|(262144,[4,5,28,3...|  2.0|
|                NSFL|               NSFL|       0|                nsfl|
[nsfl]|              [nsfl]|(262144,[13710],[...|  1.0|
|Get back to your ...|Get back to your ...|       0|get back to your ...|[get, b
ack, to, y...|[get, back, to, y...|(262144,[0,31,44,...|  1.0|
+--------------------+--------------------+--------+--------------------+-------
-------------+--------------------+--------------------+-----+
only showing top 5 rows
```

Partition Training & Test sets

In [ ]:

```
# set seed for reproducibility
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
21/12/01 20:16:00 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB

Training Dataset Count: 3891654
21/12/01 20:25:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
[Stage 41:>                                                         (0 + 1) / 1]
Test Dataset Count: 973033
```

Model Training and Evaluation

Logistic Regression using Count Vector Features

Our model will make predictions and score on the test set; we then look at the top 10 predictions from the highest probability.

In [ ]:

```
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)

predictions = lrModel.transform(testData)

# predictions.filter(predictions['prediction'] == 0).select("clean_text","catego
# .orderBy("probability", ascending=False).show(n = 10, truncate = 30)
```

```
21/12/01 20:35:31 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:45:15 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:55:36 WARN com.github.fommil.netlib.BLAS: Failed to load implementat
ion from: com.github.fommil.netlib.NativeSystemBLAS
21/12/01 20:55:37 WARN com.github.fommil.netlib.BLAS: Failed to load implementat
ion from: com.github.fommil.netlib.NativeRefBLAS
21/12/01 20:55:37 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:55:48 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:00 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:12 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:24 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:35 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:47 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:56:59 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
```

```
21/12/01 20:57:10 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:57:22 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:57:33 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:57:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:58:08 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:58:20 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:58:32 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:58:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:58:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:59:07 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:59:18 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
21/12/01 20:59:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 3.4 MiB
```

In [ ]:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
```

```
21/12/01 20:59:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 9.4 MiB
```

Out[ ]:

```
0.7704587422225653
```

Logistic Regression using TF-IDF Features

In [ ]:

```
from pyspark.ml.feature import HashingTF, IDF

hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=
idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=5) #minDocFre
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, hashingTF, idf, la

pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
```

In [ ]:

```
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(trainingData)
```

In [ ]:

```
predictions = lrModel.transform(testData)

# predictions.filter(predictions['prediction'] == 0) \
```

```
#     .select("clean_text","category","probability","label","prediction") \
#     .orderBy("probability", ascending=False) \
#     .show(n = 10, truncate = 30)
```

In [ ]:

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
lrAccuracy = evaluator.evaluate(predictions)
print(lrAccuracy)
```

Out[ ]:

```
0.7288555668318899
```

Naive Bayes

In [ ]:

```
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1)
nbModel = nb.fit(trainingData)
predictions = nbModel.transform(testData)
# predictions.filter(predictions['prediction'] == 0) \
#     .select("clean_text","category","probability","label","prediction") \
#     .orderBy("probability", ascending=False) \
#     .show(n = 10, truncate = 30)
```

In [ ]:

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
nbAccuracy = evaluator.evaluate(predictions)
print(nbAccuracy)
```

```
[Stage 103:>                                                        (0 + 1) / 1]
0.7347902005300311
```

DecisionTreeClassifier

In [ ]:

```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDep
dtModel = dt.fit(trainingData)
predictions = dtModel.transform(testData)
# predictions.filter(predictions['prediction'] == 0) \
#     .select("clean_text","category","probability","label","prediction") \
#     .orderBy("probability", ascending=False) \
#     .show(n = 10, truncate = 30)
```

In [ ]:

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
dtAccuracy = evaluator.evaluate(predictions)
print(dtAccuracy)
```

```
[Stage 115:>                                                        (0 + 1) / 1]
0.4305849554692257
```

Random Forest

In [ ]:

```
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(labelCol="label", \
                            featuresCol="features", \
                            numTrees = 100, \
                            maxDepth = 4, \
                            maxBins = 32)
# Train model with Training Data
rfModel = rf.fit(trainingData)
predictions = rfModel.transform(testData)
# predictions.filter(predictions['prediction'] == 0) \
#     .select("clean_text","category","probability","label","prediction") \
#     .orderBy("probability", ascending=False) \
#     .show(n = 10, truncate = 30)
```

```
21/12/02 01:31:08 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting lar
ge task binary with size 1059.5 KiB
```

In [ ]:

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
rfAccuracy = evaluator.evaluate(predictions)
print(rfAccuracy)
```

```
[Stage 129:>                                                        (0 + 1) / 1]
0.2570274066280338
```

OnevsRest classifer

In [ ]:

```
from pyspark.ml.classification import LogisticRegression, OneVsRest

lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)

ovr = OneVsRest(classifier=lr)

ovrModel = ovr.fit(trainingData)

# score the model on test data.
predictions = ovrModel.transform(testData)

# predictions.filter(predictions['prediction'] == 0) \
#     .select("clean_text","category","label","prediction") \
#     .show(n = 10, truncate = 30)
```

In [ ]:

```
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
ovrAccuracy = evaluator.evaluate(predictions)
print(ovrAccuracy)
```

```
[Stage 204:>                                                        (0 + 1) / 1]
0.7033108920380089
```

Visualization

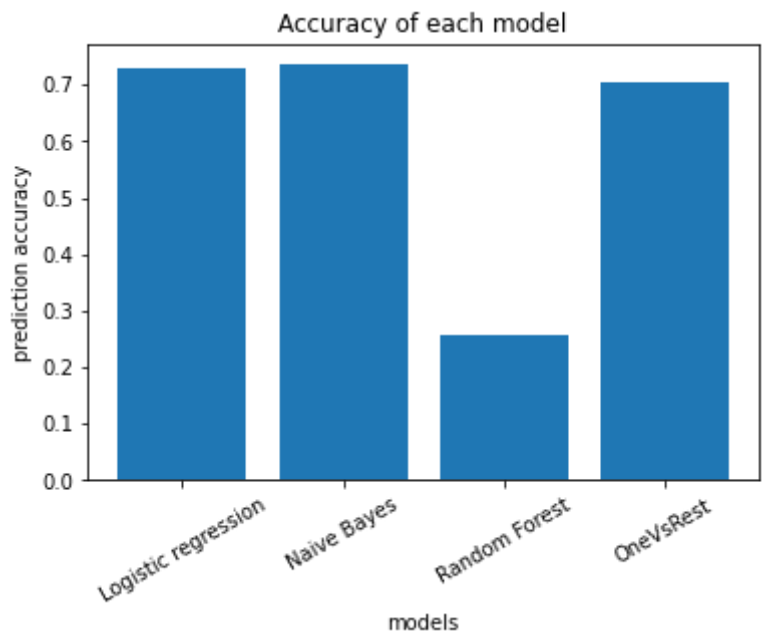
In [40]:

```
import matplotlib.pyplot as plt
import numpy as np
lrAccuracy = 0.7288555668318899
model = ['Logistic regression', 'Naive Bayes', 'Random Forest', 'OneVsRest']
accuracy = [lrAccuracy,nbAccuracy,rfAccuracy, ovrAccuracy]
```

In [41]:

```
def plot_bar_x():
    # this is for plotting purpose
    index = np.arange(len(model))
    plt.bar(index, accuracy)
    plt.xlabel('models', fontsize=10)
    plt.ylabel('prediction accuracy', fontsize=10)
    plt.xticks(index, model, fontsize=10, rotation=30)
    plt.title('Accuracy of each model')
    plt.show()

plot_bar_x()
```



In [ ]:

Cross-Validation

Let's now try cross-validation to tune our hyper parameters, and we will only tune the count vectors Logistic Regression.

In [ ]:

```
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, countVectors, labe

pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
(trainingData, testData) = dataset.randomSplit([0.8, 0.2], seed = 100)
```

```
lr = LogisticRegression(maxIter=20, regParam=0.3, elasticNetParam=0)
```

In [ ]:

```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

# Create ParamGrid for Cross Validation
paramGrid = (ParamGridBuilder()
             .addGrid(lr.regParam, [0.1, 0.3, 0.5]) # regularization parameter
             .addGrid(lr.elasticNetParam, [0.0, 0.1, 0.2]) # Elastic Net Paramet
#            .addGrid(model.maxIter, [10, 20, 50]) #Number of iterations
#            .addGrid(idf.numFeatures, [10, 100, 1000]) # Number of features
             .build())
```

In [ ]:

```
# Create 5-fold CrossValidator
cv = CrossValidator(estimator=lr, \
                    estimatorParamMaps=paramGrid, \
                    evaluator=evaluator, \
                    numFolds=5)
```

In [ ]:

```
cvModel = cv.fit(trainingData)
```

In [ ]:

```
predictions = cvModel.transform(testData)
# Evaluate best model
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(predictions)
#print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.
```

In [ ]:

count the words

In [ ]:

```
from pyspark.sql.functions import desc
# topWordsAndCountsDF = wordCount(shakeWordsDF)
topWordsAndCountsDF = wordCount(data).orderBy("count", ascending=False)
topWordsAndCountsDF.show()
```

In [ ]:

```
from pyspark.sql.functions import split, explode
#shakeWordsDF = (shakespeareDF
#                .select(explode(split(shakespeareDF.sentence, ' ')).alias('word
#                .where("word != ''"))

shakeWordsDF = (shakespeareDF
                .select(explode(split(shakespeareDF.sentence, ' ')).alias('word'
                .where(col('word') != ''))

shakeWordsDF.show()
shakeWordsDFCount = shakeWordsDF.count()
print shakeWordsDFCount
```