Yue_6k data

December 4, 2021

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

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
spark = SparkSession.builder \
    .appName("reddit_4k")\
    .getOrCreate()
sc = spark.sparkContext
```

0.1 Import Dataset

In [3]:
```
df = spark.read \
    .option("delimiter",",") \
    .option("multiLine","true") \
    .option("quote", "\"")  \
    .option("escape", "\"") \
    .option("ignoreLeadingWhiteSpace",True) \
    .csv("/user/yuewu20/data/reddit_new.csv", inferSchema=True, header=True)
```

In [4]:
```
df_nlp = df.select('id', 'subreddit', 'clean_comment', 'ups', 'downs', 'gilded', 'score
```

In [5]:
```
# import pyspark.sql.functions as f
# top_10_reddit = ['AskReddit', 'leagueoflegends','nba','funny','nfl','pics', \
#                  'videos','news', 'todayilearned','pcmasterrace']

# df_nlp = df_nlp.where(f.col('subreddit').isin(top_10_reddit)).limit(2000)
```

In [6]:
```
# df_nlp.count()
```

In [7]:
```
# df_nlp.rdd.getNumPartitions()
```

In [8]:
```
# df_nlp = df_nlp.repartition(100)
```

0.2 NLP Pipeline

In [9]:
```
from pyspark.ml import Pipeline
from pyspark.ml.feature import HashingTF, IDF, RegexTokenizer, Tokenizer, CountVectori
from pyspark.ml.regression import RandomForestRegressor, LinearRegression, DecisionTre
from pyspark.ml.evaluation import RegressionEvaluator
```

In [10]:
```
# Tokenizer
pattern = "\\W"
tokenizer = RegexTokenizer(inputCol="clean_comment", outputCol="words", pattern=patte
# tokenizer = Tokenizer(inputCol="body", outputCol="words")
df_words = tokenizer.transform(df_nlp)

# Remove stop words
remover = StopWordsRemover(inputCol="words", outputCol="filtered_words")
df_filtered = remover.transform(df_words)
```

In [11]:
```
# Select features
df_filtered = df_filtered.select('id', 'ups', 'downs', 'gilded', 'score', 'filtered_w

# Count term frequency
cv = CountVectorizer(inputCol="filtered_words", outputCol="features")
cv_model = cv.fit(df_filtered)
df_model = cv_model.transform(df_filtered)

# hashingTF = HashingTF(inputCol="filtered_words", outputCol="features", numFeatures=
# df_model = hashingTF.transform(df_filtered)
```

In [12]:
```
# # Tokenizer
# pattern = "\\W"
# tokenizer = RegexTokenizer(inputCol="body", outputCol="words", pattern=pattern)

# # Remove stop words
# remover = StopWordsRemover(inputCol="words", outputCol="filtered")

# # Count term frequency
# cv = CountVectorizer(inputCol="filtered", outputCol="rawFeatures")
# # hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=1
# idf = IDF(inputCol="rawFeatures", outputCol="features")

# # Chain indexers and forest in a Pipeline
# pipeline = Pipeline(stages=[tokenizer, remover, cv, idf])
```

In [13]:
```
# try:
#     # Load pipeline results
#     pipeline_results = joblib.load('gs://reddit-data-team-1/pipeline_results.pkl')
# except:
#     # Fit pipeline
#     model = pipeline.fit(df_nlp)
#     pipeline_results = model.transform(model)

#     # Pickle the pipeline results
#     joblib.dump(pipeline_results, 'gs://reddit-data-team-1/pipeline_results.pkl')
```

0.3 Regression Analysis

```
In [14]: # df_model = result.select('id','ups', 'downs', 'gilded', 'score', 'filtered_words',

In [15]: (training, test) = df_model.randomSplit([0.8, 0.2])
```

0.3.1 Random Forest

```
In [16]: def rf_regression(featuresCol, labelCol, training, test):

             rf = RandomForestRegressor(featuresCol=featuresCol, labelCol=labelCol)

             model = rf.fit(training)

             predictions = model.transform(test)

             return predictions
```

'ups' as label

```
In [17]: # Train model on 'ups' label
         predictions_ups = rf_regression(featuresCol="features", labelCol="ups", training=trai

         evaluator_ups = RegressionEvaluator(labelCol="ups", predictionCol="prediction", metri
         rmse_ups = evaluator_ups.evaluate(predictions_ups)
         print("Root Mean Squared Error (RMSE) on test data for 'ups' = %g" % (rmse_ups))
```

```
Root Mean Squared Error (RMSE) on test data for 'ups' = 1.25625
```

```
In [18]: predictions_ups.select('id', 'ups', 'prediction').show(10)
```

```
+-------+---+------------------+
|     id|ups|        prediction|
+-------+---+------------------+
|cqughbu|  1|0.8723324173558099|
|cqugiii|  1|0.8723324173558099|
|cqugnke|  1|0.8723324173558099|
|cqugnzs|  1|0.8723324173558099|
|cqugpu1|  1|0.8723324173558099|
|cqugtqb|  1|0.8723324173558099|
|cquh3ot|  0|0.8723324173558099|
|cquhfa9|  1|0.8723324173558099|
|cquk23e|  1|0.8723324173558099|
|cquk566|  1|0.8723324173558099|
+-------+---+------------------+
only showing top 10 rows
```

'downs' as label

```
In [19]: # Train model on 'downs' label
         predictions_downs = rf_regression(featuresCol="features", labelCol="downs", training=t

         evaluator_downs = RegressionEvaluator(labelCol="downs", predictionCol="prediction", me
         rmse_downs = evaluator_downs.evaluate(predictions_downs)
         print("Root Mean Squared Error (RMSE) on test data for 'downs' = %g" % (rmse_downs))
```

```
Root Mean Squared Error (RMSE) on test data for 'downs' = 0
```

```
In [20]: predictions_downs.select('id', 'downs', 'prediction').show(10)
```

```
+-------+-----+----------+
|     id|downs|prediction|
+-------+-----+----------+
|cqugg23|    0|       0.0|
|cqughhx|    0|       0.0|
|cqugntv|    0|       0.0|
|cqugpbm|    0|       0.0|
|cqugte7|    0|       0.0|
|cquguzk|    0|       0.0|
|cquhcwz|    0|       0.0|
|cquhkeo|    0|       0.0|
|cquk9s5|    0|       0.0|
|cqukj8g|    0|       0.0|
+-------+-----+----------+
only showing top 10 rows
```

'gilded' as label

```
In [21]: # Train model on 'gilded' label
         predictions_gilded = rf_regression(featuresCol="features", labelCol="gilded", training

         evaluator_gilded = RegressionEvaluator(labelCol="gilded", predictionCol="prediction",
         rmse_gilded = evaluator_gilded.evaluate(predictions_gilded)
         print("Root Mean Squared Error (RMSE) on test data for 'gilded' = %g" % (rmse_gilded))
```

```
Root Mean Squared Error (RMSE) on test data for 'gilded' = 0
```

```
In [22]: predictions_gilded.select('id', 'gilded', 'prediction').show(10)
```

```
+-------+------+----------+
|     id|gilded|prediction|
+-------+------+----------+
```

```
|cqughbu|     0|       0.0|
|cqugiii|     0|       0.0|
|cqugnke|     0|       0.0|
|cqugnzs|     0|       0.0|
|cqugpu1|     0|       0.0|
|cqugtqb|     0|       0.0|
|cquh3ot|     0|       0.0|
|cquhfa9|     0|       0.0|
|cquk23e|     0|       0.0|
|cquk566|     0|       0.0|
+-------+------+----------+
only showing top 10 rows
```

'score' as label

```
In [23]: # Train model on 'score' label
         predictions_score = rf_regression(featuresCol="features", labelCol="score", training=t

         evaluator_score = RegressionEvaluator(labelCol="score", predictionCol="prediction", me
         rmse_score = evaluator_score.evaluate(predictions_score)
         print("Root Mean Squared Error (RMSE) on test data for 'score' = %g" % (rmse_score))
```

```
Root Mean Squared Error (RMSE) on test data for 'score' = 1.25625
```

```
In [24]: predictions_score.select('id', 'score', 'prediction').show(10)
```

```
+-------+-----+------------------+
|     id|score|        prediction|
+-------+-----+------------------+
|cqughbu|    1|0.8723324173558099|
|cqugiii|    1|0.8723324173558099|
|cqugnke|    1|0.8723324173558099|
|cqugnzs|    1|0.8723324173558099|
|cqugpu1|    1|0.8723324173558099|
|cqugtqb|    1|0.8723324173558099|
|cquh3ot|    0|0.8723324173558099|
|cquhfa9|    1|0.8723324173558099|
|cquk23e|    1|0.8723324173558099|
|cquk566|    1|0.8723324173558099|
+-------+-----+------------------+
only showing top 10 rows
```