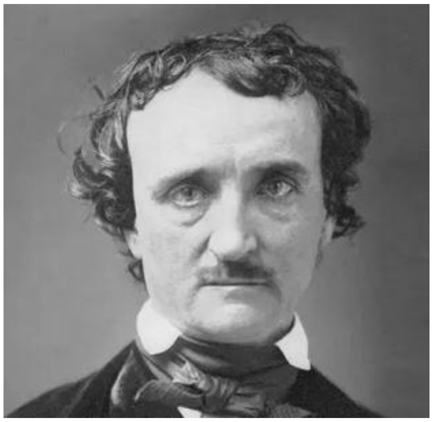
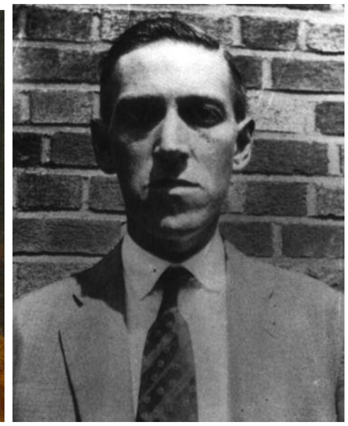
m.6.ASSIGNment → MORE Spooky authorship via Apache Spark

Overview	Apply Module 6 Spark SQL items to the spooky authorship dataset.
Data	https://www.kaggle.com/competitions/spooky-author-identification/code

?





Note: the Professor will inform you to combine or separate this new m.6.codebook with the m.5.codebook.

TASKS → <please perform all work in Apache Spark>

Task 1: Spark SQL Mechanics

1. Use spark.sql statement to join the test and train data grouping by author and ordering by id.

2. Reverse engineer so these statement run
    a. `result_df = spark.sql("""`
    b. `    SELECT text AS sentence,`
    c. `        size(split(text, ' ')) AS word_count`
    d. `    FROM spooky_sentences`
    e. `    ORDER BY word_count DESC`
    f. `""")`
    g. `# Show the resulting DataFrame`
    h. `result_df.show(10, truncate=False)`

3. Write a subquery to count total words by author generating something like
    a. `+------------------+-----------+`
    b. `| author           | total_words|`
    c. `+------------------+-----------+`
    d. `| Edgar Allan Poe  | 23456     |`
    e. `| H.P. Lovecraft   | 19876     |`
    f. `| Mary Shelley     | 17654     |`
    g. `+------------------+-----------+`

Task 2: Data Loading and Query Types

4. Write a user-defined function classifying word count >30 as wordy, <7 words as pity, and the difference as not wordy generating an outcome something like

```
a. +-----------------+-----------------+
b. |    author       | author_category |
c. +-----------------+-----------------+
d. | Edgar Allan Poe |    not wordy    |
e. | H.P. Lovecraft  |      wordy      |
f. | Mary Shelley    |    not wordy    |
g. +-----------------+-----------------+
```

Task 3: Advanced SQL Functions and Expressions

5. Use functions “lower” and “concat” to combine all sentences into one string displaying something like

```
a. +---+-----------------------------------------------------------------+
b. |id |author           |concatenated_text                              |
c. +---+-----------------------------------------------------------------+
d. |1  |Edgar Allan Poe  |1 this is a sample sentence                    |
e. |2  |H.P. Lovecraft   |2 this is another sample sentence              |
f. |3  |Mary Shelley     |3 and another sentence                               |
```

Task 4: Views and Temporary Tables

6. Create a view using spark.sql to displaying any one sentence for each author with words >30.

```
a. +-----------------+-------------------------------------------------------------------------+
b. |author           |text                                                                     |
c. +-----------------+-------------------------------------------------------------------------+
d. |Edgar Allan Poe  |In the greenest of our valleys,By good angels tenanted,Once a fair   |
e. |H.P. Lovecraft   |Life and death were not explicit alternatives. It was a gray world   |
f. |Mary Shelley     |The road ran by the side of the sea wall, along the brow of the      |
g. +-----------------+-------------------------------------------------------------------------+
```

Task 5: Error Handling and Debugging

7. Add to task.6 a “try-except” block for any item you chose as long as its valid. For instance, it could be an error for reading a file, displaying an entry without a sentence and similar.

```
a. print("Error reading the file: File not found.")
b. print("Error occurred displaying entry without a sentence: KeyError: 'text'")
```

Task 6: Spark SQL for Machine Learning

8. Calculate the lexical density by author displaying something like

```
a. +-----------------+------------------+
b. |           author|  lexical_density |
c. +-----------------+------------------+
d. |  Edgar Allan Poe| 0.07851234562311 |
e. |   H.P. Lovecraft| 0.08474362093088 |
f. |     Mary Shelley| 0.07691248914225 |
```

m.5.ASSIGNment → Spooky authorship via Apache Spark

DETAILS

Dataset Description: The spooky author identification dataset contains text from works of fiction written by spooky authors of the public domain: Edgar Allan Poe, HP Lovecraft and Mary Shelley. The data was prepared by chunking larger texts into sentences using CoreNLP's MaxEnt sentence tokenizer resulting in an odd non-sentence here and there. Your objective is to accurately identify the author of the sentences in the test set.

- **id** - unique identifier for each sentence
- **text** - sentence written by one of the authors
- **author** - {EAP:Edgar Allan Poe}, {HPL:HP Lovecraft}; {MWS:Mary Wollstonecraft Shelley}

Objective:

A. Accurately identify the author of the sentences in the test set.
B. Perform ALL work using Apache Spark.

Dataset:

- Training consists of passages with an author label.
- Test has sentences with no author labels.

Competition Evaluation: The submissions were evaluated based on multi-class logarithmic loss. The logarithmic loss assesses the uncertainty of the predicted probabilities, penalizing confident incorrect predictions. Lower log loss values indicated better performance.

Approach: NLP techniques + machine learning algorithms. Feature engineering like bag-of-words, TF-IDF, word embeddings/Word2Vec. Perform algorithmic work with logistic regression, support vector machines, neural networks, and as appropriate.

TASKS

Stage 0: Import Data

1. Create a code notebook called: code_6_of_10_data_mine_<your_name>.ipynb
2. Load data into Spark data objects and explore structure, size, and distribution of information.

Stage 1: Data Preparation - Exploratory data analysis and text mining pre-processing

3. Perform exploratory data analysis and create visualizations and tables as needed.
4. Text Preprocessing: perform tasks like tokenization and stopwords removal to clean text data.
    a. Tokenize - split the text into individual words aka tokens.
    b. Remove stop.words - frequently used pronouns and personal references.
        i. Top ten include: I, you, he, she, it, we, they, me, him, her
    c. Lemmatization - convert words to their root (optional).
        i. Lemmatization is a text normalization technique that reduces words to their base or dictionary form (lemma). Use to reduce inflected or derived words to their root form for better analysis and modeling outcomes.

```
from pyspark.ml.feature import StopWordsRemover, Tokenizer
from pyspark.ml.feature import CountVectorizer, IDF
from pyspark.ml.feature import Normalizer
from pyspark.ml import Pipeline
```

```
# Step 1: Tokenization
tokenizer = Tokenizer(inputCol="text", outputCol="tokens")
# Step 2: Stop word removal
stopwords_remover = StopWordsRemover(inputCol="tokens", outputCol="filtered_tokens")
'''
Step.1 replace "text" in Tokenizer(inputCol="text", outputCol="tokens")
with the actual column name from your CSV file that contains the text data.
'''
```

Stage 2: Feature Extraction

5. Perform TF-IDF to quantify word importance <term.frequency.inverse.doc.frequency>
6. Normalize is scaling or standardizing the numerical features to a standard range or distribution.
    a. In text mining, normalization vectorizes features with methods like TF-IDF, a numerical measurement, to ensure a consistent scale.
    b. It handles variations in the magnitude of feature values impacting machine-learning algorithm performance. Normalize the features to ensure a similar scale and prevent features with larger values from dominating the analysis or modeling process.

```
# Step 3: TF-IDF calculation
vectorizer = CountVectorizer(inputCol="filtered_tokens", outputCol="vectorized_tokens")
idf = IDF(inputCol="vectorized_tokens", outputCol="tfidf")
# Step 4: Normalization
normalizer = Normalizer(inputCol="tfidf", outputCol="normalized_features")
# Step 5: Create pipeline for chaining the text mining transformers
pipeline = Pipeline(stages=[tokenizer, stopwords_remover, vectorizer, idf, normalizer])
# Step 6: Apply the pipeline to DataFrame
processed_data = pipeline.fit(your_dataframe).transform(your_dataframe)
'''
step.4 The processed_data object will contain the final processed features in the
"normalized_features" column. use for machine learning tasks.

Step.6 replace your_dataframe with the name of your DataFrame that holds the CSV data.
'''
```

TASKS (continue)

Stage 3: Machine Learning

7. Perform train\test split.
8. Perform algorithmic analysis to assess and predict test labels.
    a. Use as many algorithms as you need to get a good answer.
    b. Supervised: logistic regression, random forest, support vector machines, etc.
    c. Unsupervised: K-means, dimensionality reduction, PCA, etc.

Stage 4: Evaluation & Visualization

9. Choose a metric strategy to assess algorithmic performance like accuracy, precision, recall, or F1 score.
10. Visualize confusion matrix, correlations, and similar.
11. Identify important features contributing to classification.
12. Write a 2-3 sentence minimum of findings, learnings, and what you would do next.

```
from pyspark.ml import Pipeline
from pyspark.ml.feature import StopWordsRemover, Tokenizer, CountVectorizer, IDF
```

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

1. Tokenizer to split split part of the pyspark.ml.feature module
2. StopWordsRemover in the pyspark.ml.feature module.
3. lemmatization - PySpark does not have a built-in lemmatization use User-Defined Functions UDF to create.
4. normalize like convert text to lowercase; remove special characters with PySpark user defined functions <UDF> or built-in string functions like lower() or regexp_replace().
5. tf.idf PySpark's CountVectorizer and IDF class compute tf.idf for text feature extraction.

Resources

- algorithmic assessment reference table
- PySpark Code Dictionary
- pd.DataFrame vs. rdd.DataFrame
- Cheat Sheet for PySpark
- Additional PySpark Guide
- Cheat Sheet for LaTeX
  How to Assess Algorithm Fit
- Apache Spark Source Documentation
- Feng, W. (2021). *Learning Apache Spark with Python*. GitHub.

Additional Resources

- Feng, W., & Chen, M. (2017). *Learning Apache Spark*. GitHub.
- Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning Spark: Lightning-fast big data analysis*. O'Reilly Media, Inc.
- Kirillov, A. (2016). *Apache Spark: Core concepts, architecture and internals*. Datastrophic.