TEXT CATEGORIZATION USING SAGEMAKER

Project Report

Aakef Waris (A20420535)
Amandeep Singh Oberoi (A20466752)
Amit Nikam (A20470263)
Rutul Mehta (A20476293)

Table of Contents

1. Abstract

As a part of understanding the workings of the SageMaker, we implement a practice project on sagemaker prior to our actual project implementation. In this practice project we used the follow source as base [1] and executed sentiment analysis and text categorization on Amazon Review dataset available publicly. This dataset was clean and easy to work on. Once we knew how sagemaker worked, we started the working on the following project.

The core problem that is being solved is stance classification in our main project. The training set of data is composed of article headlines. The headline either reports on covid spread diminishing (pro-mitigation) or increasing (anti-mitigation), or it may be unrelated (unclear). The labels are as stated: [pro-mitigation, anti-mitigation, unclear] This leaves us with a multiclass classification problem. From here we decided to explore AWS sagemaker to host jupyter notebook instances and train models. The data was manually labeled from students participating in the Fall 2021 CS585 Natural Language Processing course at the Illinois Institute of Technology. A series of classification models were used to learn from features from the text and predict the labels.

Due to cost constraint, out of 3 models, we have implemented only 1 model in the second project.

2. Contribution

2.1 Aakef Waris

Data collection, review various datasets from Kaggle and AWS, find reliable dataset based-on our use case, feature engineering, model implementation approach, model evaluation, code optimization, report writing.

2.2 Amandeep Singh Oberoi

Data collection, review various datasets from Kaggle, AWS, find reliable dataset based-on our use case, data visualization, model implementation approach, implementation of the model, model evaluation, report writing, final report formatting

2.3 Amit Nikam

Data visualization, data processing, feature engineering, model implementation approach, implementation of the model, model evaluation, code optimization, report writing, final report formatting.

2.4 Rutul Mehta

Literature review, comparative analysis of various ML systems, data processing, feature engineering, model implementation approach, implementation of the model, model evaluation, report writing, final report formatting.

2.5 Project Implementation approach

- As a part of a data collection, our first step is to identify reliable and suitable dataset according to our use case. We took a group call and found various online datasets for amazon review system. We have reviewed some of the datasets from AWS [2] but most of them are them are irrelevant and some of them are big in size which will increase our costing. Then decided to use dataset which is suitable for sentiment analysis and small in size.
- Then we move to the next step, we performed data processing part. In data processing, we implemented several things like keep only text with English word, remove punctuation, tokenizing word, remove stop words etc. Data processing part took some time and efforts because this dataset is retrieved from raw source is was never worked on before.
- Then we did some interesting feature engineering to find patterns and relations within the data so that that could be used within the models for training to improve the accuracy slightly.
- After that we wanted to implement some models on our datasets. For that we took a call to figured out which classification and regression model is suitable for our dataset. Implementation part required collaborative efforts.
- Finding the accuracy from different evaluation matrix is straight forward. From confusion matrix we found F1, recall and precision score.

- Lastly, we wrote report in proper format. This required combine efforts.

3. Literature review

3.1 Amazon sagemaker

Amazon SageMaker is an excellent tool for creating machine learning models that require more effort than a simple point-and-click analysis. SageMaker integrates nicely with other Amazon technologies, so if you use or are considering using Amazon Web Services, SageMaker would be a terrific addition. SageMaker is good for consumer insights, predictive analytics, and uncovering hidden gems in the huge amounts of data we generate [3]. SageMaker is not well suited to analyze small data.

3.2 Spark

On single-node workstations or clusters, Apache SparkTM is a multi-language engine for data engineering, data science, and machine learning [4]. SQL, machine learning, graph computation, and stream processing are all libraries that can be utilized together in an application. Java, Python, Scala, and R are among the programming languages supported by Spark. Spark allows application developers and data scientists to query, analyze, and transform data at scale quickly and easily. ETL and SQL batch workloads across massive data sets, streaming data processing from sensors, IoT, or financial systems, and machine learning tasks are the most common tasks linked with Spark [5].

3.2.1 Spark MLlib

In Apache Spark, MLlib is utilized to do machine learning. MLlib is a collection of well-known algorithms and tools which is a scalable Machine Learning library that considers both high-quality algorithms and fast performance. Regression, classification, clustering, pattern mining, and collaborative filtering are examples of machine learning algorithms we can implement. MLlib also includes lower-level machine learning primitives such as the general gradient descent optimization technique.

The framework offers [6]:

- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

3.3 H2O

H2O is a distributed in-memory machine learning platform with linear scalability that is open-source. H2O features an AutoML feature and supports the most widely used statistical and machine learning methods [7]. H2O works with large data technologies like Hadoop and Spark and leverages familiar interfaces like R, Python, Scala, Java, JSON, and the Flow notebook/web interface. Through faster and better predictive modeling, H2O can easily and quickly generate insights from data [7].

3.4 TenserFlow

TensorFlow is an open-source end-to-end machine learning platform. It's a symbolic math toolkit that employs dataflow and differentiable programming to handle a variety of tasks related to deep neural network training and inference. It enables programmers to design machine learning applications utilizing a variety of tools, frameworks, and open-source resources. Google's TensorFlow is now the most well-known deep learning package on the planet. Machine learning is used by Google in all of its products to enhance search, translation, picture captioning, and recommendations.

Tensorflow gets its name from its underlying framework, Tensor. Tensors are used in every computation of Tensorflow. A tensor is an n-dimensional vector or matrix that may represent any form of data. A tensor's values all have the same data type with a known (or partially known) form. The dimensionality of the matrix or array determines the form of the data.

It's portable, since the graph may be used right away or saved for later use, and it runs on a variety of platforms, including CPUs, GPUs, TPUs, smartphones, and embedded. It may also be delivered to production without relying on any of the code that created the graph, merely the runtime required to run it.

It's transformable and optimizable, since the graph may be changed to provide a better version for a particular platform. Also possible are memory and computation optimizations, as well as trade-offs between the two. This is important for providing quicker mobile inference following training on bigger computers, for example. Distributed execution is supported as well [8].

3.5 Comparison

When implementing our project we had a variety of options primarily: AWS Sagemaker, H20.ai H20, Apache SparkMLib and TenserFlow. All four are very popular large scale ML tools, despite providing some different benefits. Both Sagemaker and H20 are AutoML tools. AutoML tools provide a suite of cloud ML services that automate as much of the machine learning pipeline as possible. SparkMLib is simply a machine learning library built on top of Apache Spark and it allows developers to run common machine learning algorithms on large, distributed datasets typically stored in Spark Dataframes or in a distributed file system like HDFS. The primary difference here is that SparkMLib is just a library that is meant to be run on distributed systems. Where Sagemaker and H20 are cloud platforms offering automated services.

Table 1 Advantage and disadvantage of ML system

ML Systems	**Advantage**	**Disadvantage**
Amazon Sagemaker	❖ Easy model creation without deep knowledge of ML. ❖ Customization and easy to alter and change. ❖ No need of additional software. ❖ Easy deployment of the model. ❖ Good documentation	❖ Although most of the popular library and machine learning frameworks are present, we must continue to rely on them for new releases. ❖ Lacking some machine learning pipelines
Spark MLlib	❖ Ease of use ❖ Dynamic in nature ❖ Various framework offers for Featurization, ML algorithm, pipeline	❖ Does not suit for multi-user environment ❖ File management system. ❖ Less algorithm compared other ML system
H2O	❖ Faster than Python scikit learn ❖ It can be accessed from java as well as python ❖ You can start with the clickable interface and then move to the python console ❖ Open source	❖ No containerization facility like docker should be given ❖ No better documentation ❖ Need to improve visual presentation
TensorFlow	❖ A vast library of functions for all kinds of tasks - Text, Images, Tabular, Video etc. ❖ The use of high-level libraries like Keras and Estimators makes it very easy for a beginner to get started with neural network models.	❖ Cannot handle sequence inputs ❖ RNNs are still a bit lacking, compared to Theano. ❖ Computation speed ❖ Missing symbolic loops

4. Methodology

4.1 Data storage and Collection

We implemented public storage cloud resource for storing objects known as S3. Amazon S3 buckets are comparable to file folders in that they hold objects that include data and descriptive information.

AWS offers this service to store and protect data for a range of use cases like Backup and restore, archiving, corporate applications, IoT devices, and big data analytics. Consumers may use Amazon S3's administration tools to optimize, organize, and customize data access to meet your unique business, organizational, and compliance needs [9].

We used Boto3 and created a session of S3 using current region (us-east-2): The Python SDK for AWS is known as Boto3. It allows you to create, update, and remove Amazon Web Services resources directly from Python scripts.

Data saved to Directory

- Generating file list from bucket
- Uploading all files from S3 to Sagemaker notebook

We are using a crowdsourced data collected from NY times and Change.org. Dataset talks about general public's opinion about Covid-19, vaccination, masks, lockdowns and government policies.

Data is self-annotated with labels

- **Pro-mitigation**: Writer is more inclined that masks, lockdown and policies can mitigate Covid-19.
- **Anti-mitigation**: Writer is against the lockdown in schools & business, effectiveness of vaccination on Covid 19 and believes that government measures won't be effective towards the situation
- **Unclear**: When stance is unclear or write is off topic. Sometimes having both positive and negative stance in same line can make label unclear.

4.2 Data processing

- Replace **nan values** with string "missing"
- Stratifying dataset distribution into "train" and "test" based on labels, so that we can work with balanced distribution to avoid "**Classifier bias**" caused due to unbalanced classes.
- To perform Modelling of our dataset:
    - Generating **bag of words** representation for lexical analysis: Bag of words uses Count vectorizers for each document. We made use of sklearn's package: Textual Feature Extraction module. There are a total of 7033 words in our vector representation.
- Data uses canonical forms of same stem word and indicates presence of other languages as well.
- Bag of words suffers from one problem: It gives extra weight to high frequency words which might not contribute much to our model. Using **XOR operator** we generated a list of unique words which might be more effective in training.
- **Function preprocess**:
    - Uses lower case (make model case insensitive)
    - Remove special characters, emoticons, punctuations, numbers
    - Tokenize words
    - Use lemma inflections instead of whole words: Use one stem for all canonical forms
    - Remove stop words: words which don't contribute much to model (this, a, the etc.)

4.3 Feature Engineering

- Annotation Score: **Cohen Kappa:**
    - To understand and analyses annotation (labelling) accuracy: we generated scores for each data frame and removing all fields with annotator agreement score of less than 0.2
- **Sentiment analysis:**

    - Generating polarity scores between two labels, the labels are subjective which means opposite labels should have opposite polarity which we can use as a feature in our model.
- **Language modelling**:
    - Single words usually don't tell us much information about context: Ex- "New" and "York" might not be very informative separately, but "NEW YORK" together might tell us more about context.
    - Using Frequently used Bigram (Two-word phrases) which are mostly used as features can help improve our model.
- **Word Count**:
    - Average word count for each text for each class could be used as an extra feature.

4.4 Model comparison

4.4.1 Logistic Regression

For classification, Logistic Regression is one of the simplest and widely used Machine Learning methods. It's simple to set up and may be used as a starting point for any binary classification task. Deep learning can benefit from its core foundational notions. The link between one dependent binary variable and independent factors is described and estimated using logistic regression.

Assumption for Logistic regression:

Logistic regression assumes that independent variables are linear and that log odds are log(p/(1-p)), where p is the likelihood of success.

There must be minimal or no multicollinearity among the independent variables for logistic regression to work. This implies that the independent variables should have a low correlation with one another.

Activation: Sigmoid function, also called logistic function maps a value from 0 to 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of spam in email.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as bars or movie rating from 1 to 5.

Advantage:

- Less inclined to overfitting
- Easily extendable to multiple classes
- Less training required and easy to understand

Disadvantages:

- Non linearity is an issue
- Correlation of features affects performance
- Feature engineering necessary step

4.4.2 Gaussian naïve Bayes

The Bayes theorem provides the basis for a collection of supervised machine learning classification algorithms known as Naive Bayes. It's a basic categorization approach with a high degree of success. They're useful when the inputs' dimensionality is high. The Naive Bayes Classifier may also be used to solve complex classification issues.

Bayes theorem, which is the basis for this algorithm:

$$p(c|x) = p(x|c)\frac{p(c)}{p(x)}$$

Assumption for Gaussian naïve bayes

The high independence assumptions between the characteristics are one of the assumptions made. These classifiers make the assumption that the value of one feature is unrelated to the value of any other characteristic.

When working with continuous data, one common assumption is that the continuous values associated with each class follow a normal (or Gaussian) distribution.

Advantages:

- Works well with less data
- performs well in case of text analytics problems

Disadvantages:

- Relies on and often an incorrect assumption of independent features
- Not ideal for data sets with a large number of numerical attributes
- Suffers from phenomenon referred to as ‘Zero frequency’

4.4.3 Randomforest

A random forest is a machine learning approach for solving classification and regression issues. It makes use of ensemble learning, which is a technique for solving complicated problems by combining several classifiers.

Many decision trees make up a random forest algorithm. Bagging or bootstrap aggregation are used to train the 'forest' formed by the random forest method. Bagging is a meta-algorithm that increases the accuracy of machine learning methods by grouping them together.

Because each decision tree is fit on a little different training dataset, and hence has a slightly varied performance, tagging is an excellent ensemble approach. Unlike traditional decision tree models like classification and regression trees (CART), trees in the ensemble are not pruned, resulting in minor overfitting to the training dataset. This is advantageous because it allows each tree to be more distinct, resulting in fewer associated predictions or prediction mistakes.

Advantages:

- Easy Data Preparation
- Fast Training
- Suitable for large

Disadvantages:

- Overfitting Risk
- Parameter Complexity for tuning and optimization

4.5 Evaluation Matrix

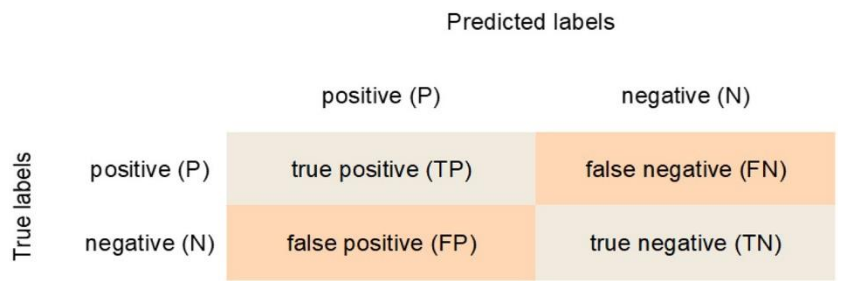
4.5.1 Confusion matrix:

It is a tabular summary of the number of correct and incorrect predictions made by a classifier. Here, row and column represent the actual label and predicted label respectively.

It can be used to evaluate the performance of a classification model through the calculation of performance metrics like accuracy, precision, recall, and F1-score. This matrix is not only useful to analyze the performance but also capable to measure the error in the model.

4.5.2 Accuracy, precision, recall, and F1 score

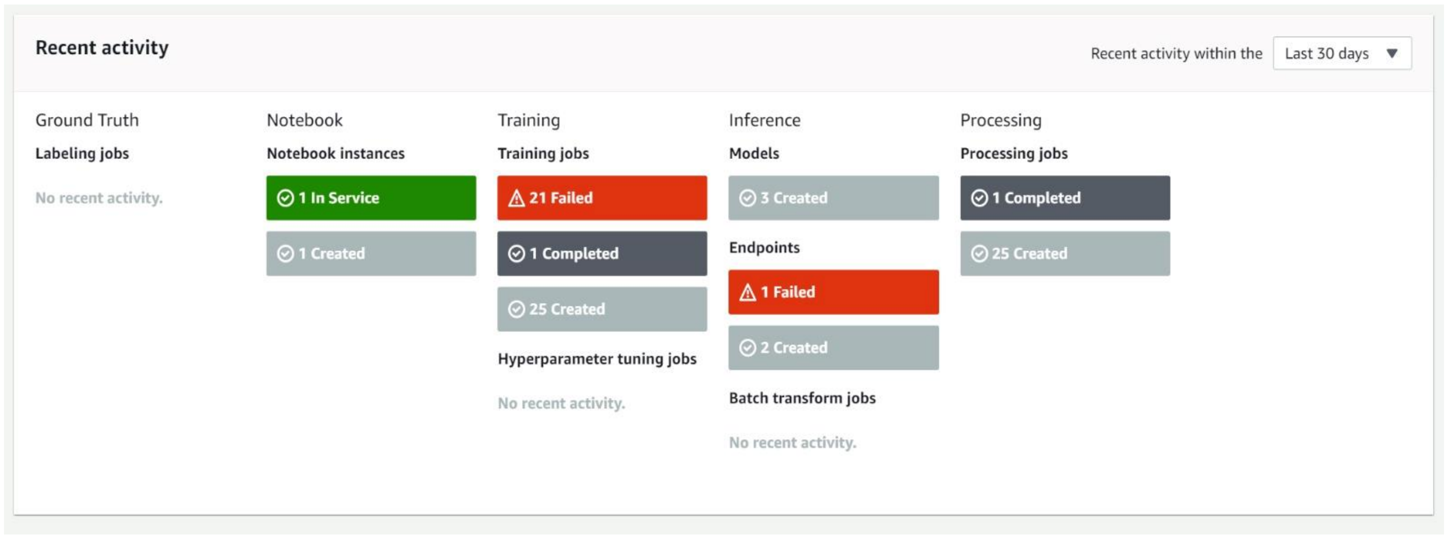
These evaluation scores are measured from the component of the confusion matrix (True positive (TP), False negative (FN), False positive (FP), True negative (TN)). Datasets with high polarity or unbalanced classes cannot be evaluated through simple accuracy but precision, recall, and F1 score are needed to sufficient to handle such datasets.



		Predicted labels	
		positive (P)	negative (N)
True labels	positive (P)	true positive (TP)	false negative (FN)
	negative (N)	false positive (FP)	true negative (TN)

Figure 1 Confusion matrix [10]

5. Implementation Screenshot



Recent activity

Recent activity within the Last 30 days

Ground Truth
Labeling jobs
No recent activity.

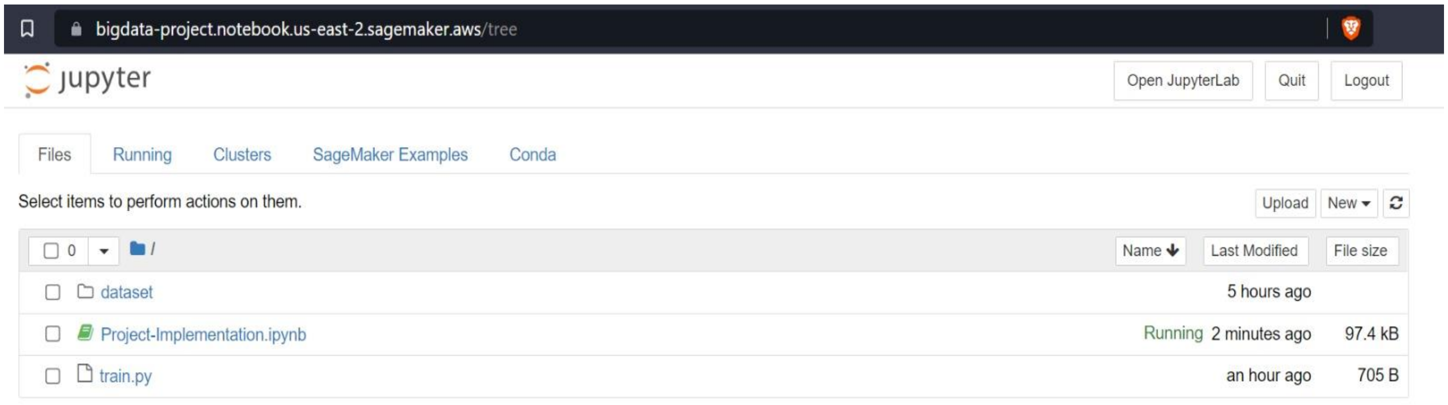
Notebook
Notebook instances
1 In Service
1 Created

Training
Training jobs
21 Failed
1 Completed
25 Created
Hyperparameter tuning jobs
No recent activity.

Inference
Models
3 Created
Endpoints
1 Failed
2 Created
Batch transform jobs
No recent activity.

Processing
Processing jobs
1 Completed
25 Created

Figure 2 SageMaker dashboard

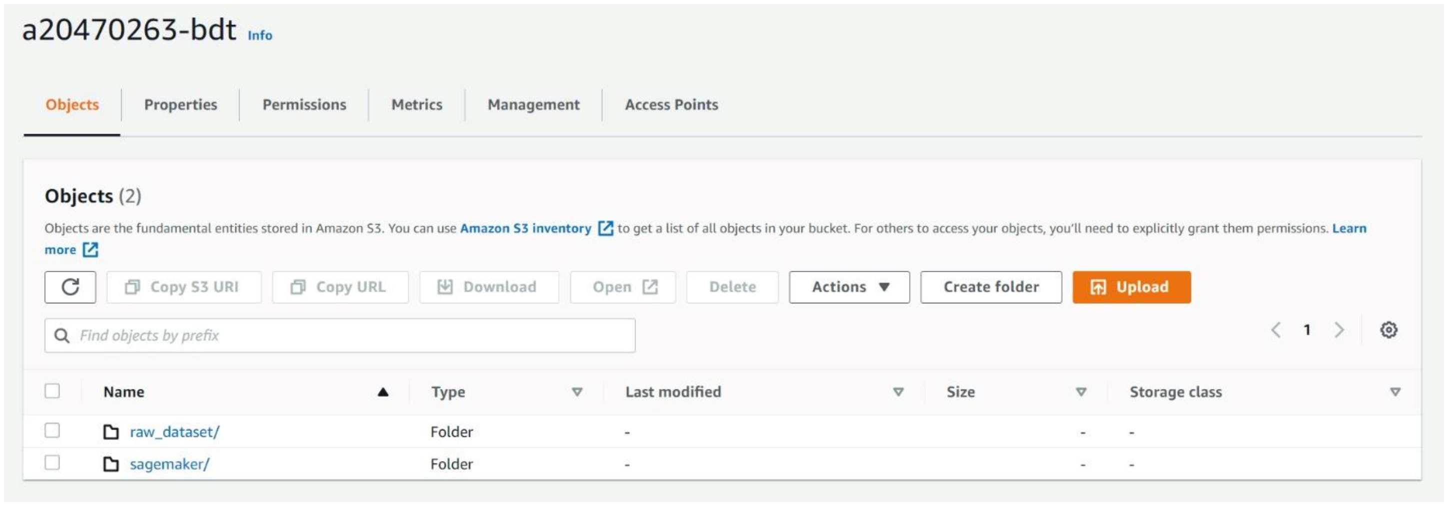


bigdata-project.notebook.us-east-2.sagemaker.aws/tree

Select items to perform actions on them.

Name		Last Modified	File size
dataset		5 hours ago	
Project-Implementation.ipynb	Running	2 minutes ago	97.4 kB
train.py		an hour ago	705 B

Figure 3 SageMaker Jupyter notebook ml.t2.medium Instance



a20470263-bdt Info

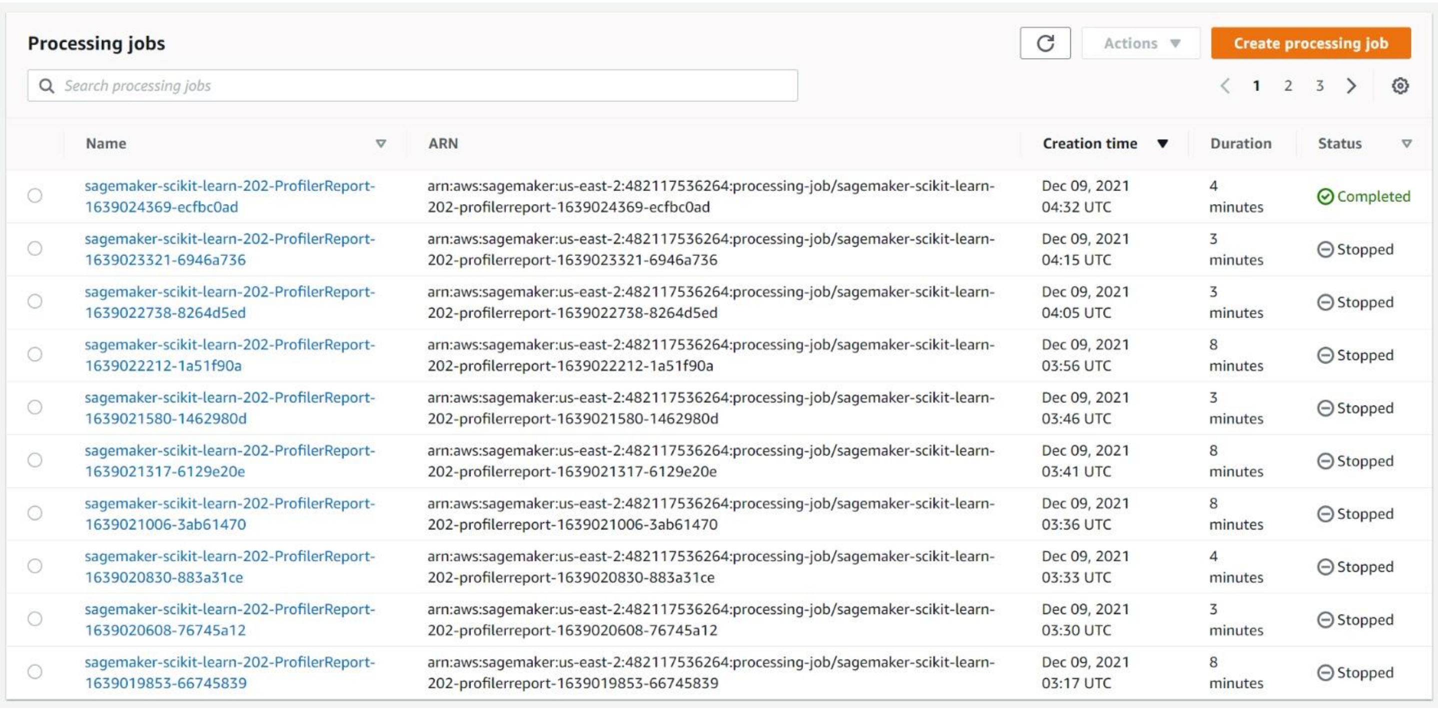
Objects | Properties | Permissions | Metrics | Management | Access Points

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use **Amazon S3 inventory** to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. **Learn more**

Name	Type	Last modified	Size	Storage class
raw_dataset/	Folder	-	-	-
sagemaker/	Folder	-	-	-

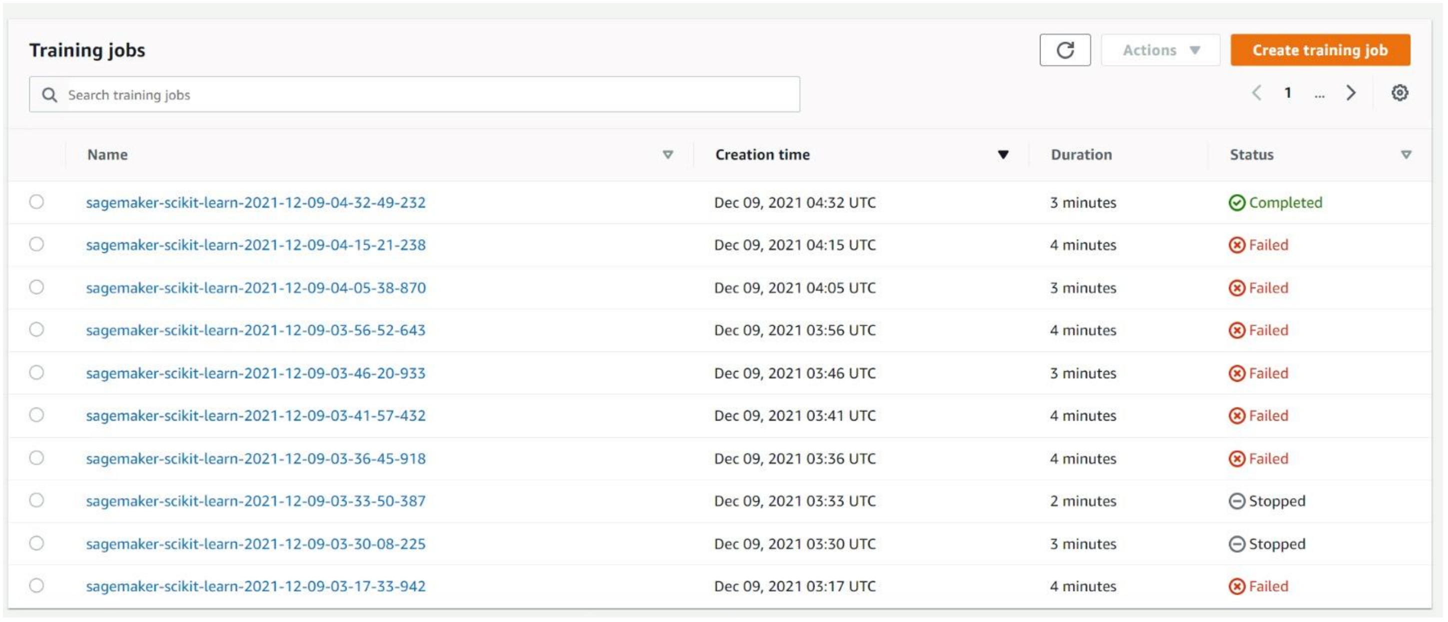
Figure 4 s3 bucket



Processing jobs

Name	ARN	Creation time	Duration	Status
sagemaker-scikit-learn-202-ProfilerReport-1639024369-ecfbc0ad	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639024369-ecfbc0ad	Dec 09, 2021 04:32 UTC	4 minutes	Completed
sagemaker-scikit-learn-202-ProfilerReport-1639023321-6946a736	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639023321-6946a736	Dec 09, 2021 04:15 UTC	3 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639022738-8264d5ed	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639022738-8264d5ed	Dec 09, 2021 04:05 UTC	3 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639022212-1a51f90a	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639022212-1a51f90a	Dec 09, 2021 03:56 UTC	8 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639021580-1462980d	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639021580-1462980d	Dec 09, 2021 03:46 UTC	3 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639021317-6129e20e	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639021317-6129e20e	Dec 09, 2021 03:41 UTC	8 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639021006-3ab61470	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639021006-3ab61470	Dec 09, 2021 03:36 UTC	8 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639020830-883a31ce	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639020830-883a31ce	Dec 09, 2021 03:33 UTC	4 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639020608-76745a12	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639020608-76745a12	Dec 09, 2021 03:30 UTC	3 minutes	Stopped
sagemaker-scikit-learn-202-ProfilerReport-1639019853-66745839	arn:aws:sagemaker:us-east-2:482117536264:processing-job/sagemaker-scikit-learn-202-profilerreport-1639019853-66745839	Dec 09, 2021 03:17 UTC	8 minutes	Stopped

Figure 5 Processing Jobs we tried



Training jobs

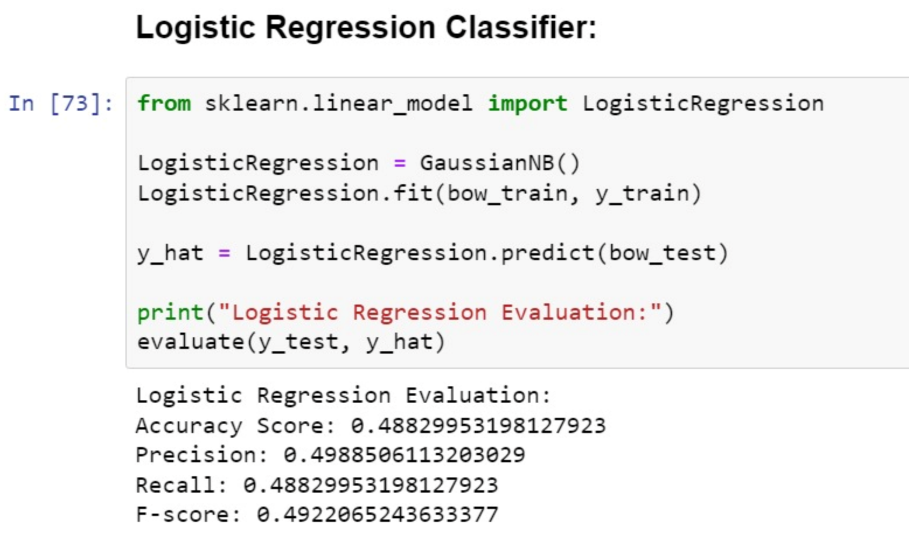
Search training jobs

Actions | Create training job

Name	Creation time	Duration	Status
sagemaker-scikit-learn-2021-12-09-04-32-49-232	Dec 09, 2021 04:32 UTC	3 minutes	Completed
sagemaker-scikit-learn-2021-12-09-04-15-21-238	Dec 09, 2021 04:15 UTC	4 minutes	Failed
sagemaker-scikit-learn-2021-12-09-04-05-38-870	Dec 09, 2021 04:05 UTC	3 minutes	Failed
sagemaker-scikit-learn-2021-12-09-03-56-52-643	Dec 09, 2021 03:56 UTC	4 minutes	Failed
sagemaker-scikit-learn-2021-12-09-03-46-20-933	Dec 09, 2021 03:46 UTC	3 minutes	Failed
sagemaker-scikit-learn-2021-12-09-03-41-57-432	Dec 09, 2021 03:41 UTC	4 minutes	Failed
sagemaker-scikit-learn-2021-12-09-03-36-45-918	Dec 09, 2021 03:36 UTC	4 minutes	Failed
sagemaker-scikit-learn-2021-12-09-03-33-50-387	Dec 09, 2021 03:33 UTC	2 minutes	Stopped
sagemaker-scikit-learn-2021-12-09-03-30-08-225	Dec 09, 2021 03:30 UTC	3 minutes	Stopped
sagemaker-scikit-learn-2021-12-09-03-17-33-942	Dec 09, 2021 03:17 UTC	4 minutes	Failed

Figure 6 Training Jobs we tried

6. Project Evaluation



Logistic Regression Classifier:

In [73]:
```python
from sklearn.linear_model import LogisticRegression

LogisticRegression = GaussianNB()
LogisticRegression.fit(bow_train, y_train)

y_hat = LogisticRegression.predict(bow_test)

print("Logistic Regression Evaluation:")
evaluate(y_test, y_hat)
```

```
Logistic Regression Evaluation:
Accuracy Score: 0.48829953198127923
Precision: 0.4988506113203029
Recall: 0.48829953198127923
F-score: 0.4922065243633377
```

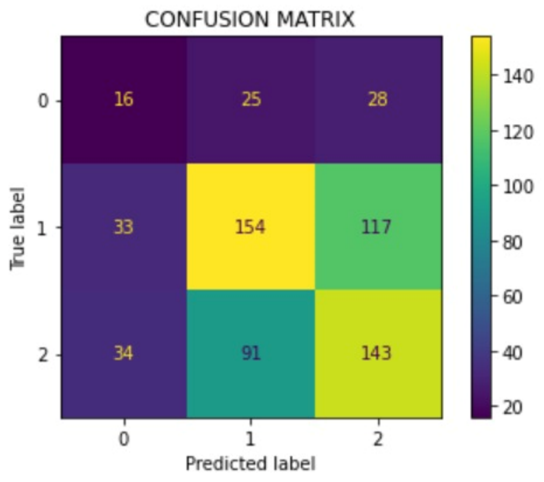


Figure 7 Evaluation for Logistic Regression

Gaussian NB:

This model is created in notebook directly, rather

than sagemaker's container due to cost constraint.

```
In [70]: from sklearn.naive_bayes import GaussianNB

         GaussianNBdel = GaussianNB()
         GaussianNBdel.fit(bow_train, y_train)

         y_hat = GaussianNBdel.predict(bow_test)

         print("GaussianNB Evaluation:")
         evaluate(y_test, y_hat)
```

```
GaussianNB Evaluation:
Accuracy Score: 0.48829953198127923
Precision: 0.4988506113203029
Recall: 0.48829953198127923
F-score: 0.4922065243633377
```

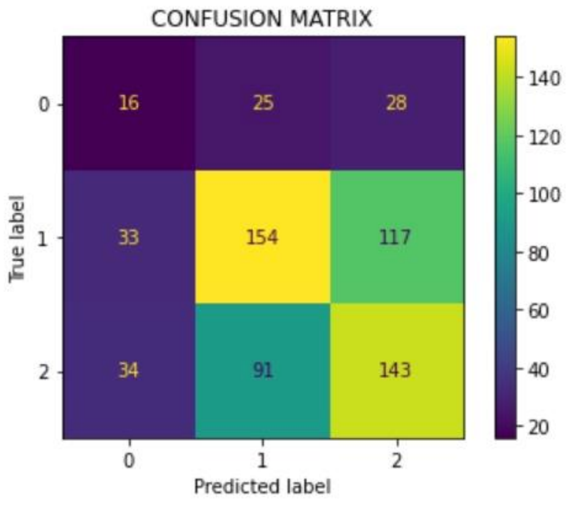
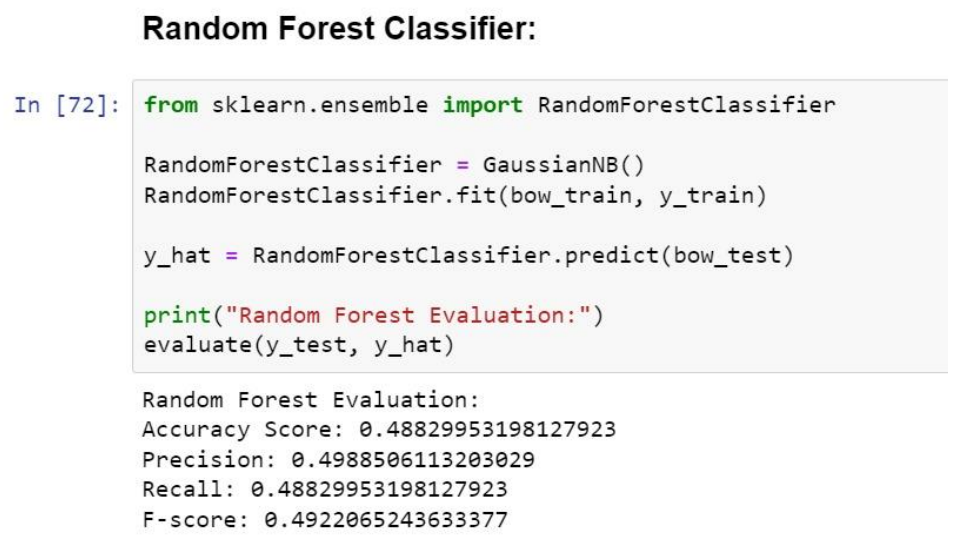


Figure 8 Evaluation of GaussianNB



Random Forest Classifier:

```
In [72]: from sklearn.ensemble import RandomForestClassifier

         RandomForestClassifier = GaussianNB()
         RandomForestClassifier.fit(bow_train, y_train)

         y_hat = RandomForestClassifier.predict(bow_test)

         print("Random Forest Evaluation:")
         evaluate(y_test, y_hat)
```

```
Random Forest Evaluation:
Accuracy Score: 0.48829953198127923
Precision: 0.4988506113203029
Recall: 0.48829953198127923
F-score: 0.4922065243633377
```

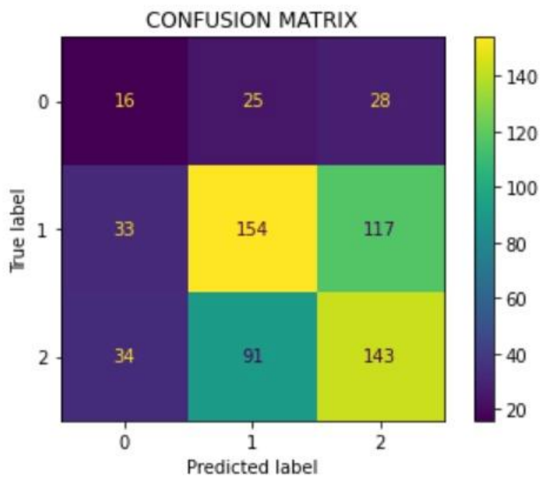


Figure 9 Evaluation of Random Forest Classifier

7. Conclusion

We implemented two different projects using SageMake. Through both of these projects we learnt bits and nuts of implementing machine learning project in Sagemaker which can also work with any big size datasets. Thank you for giving such opportunity to implement such an industry level project.

8. Reference

[1] "Amazon SageMaker Example for NLP Tasks." https://sagemaker-examples.readthedocs.io/en/latest/use-cases/product_ratings_with_pipelines/pipelines_product_ratings.html (accessed.

[2] "Registry of open data on AWS." https://registry.opendata.aws/ (accessed.

[3] "Amazon SageMaker." https://en.wikipedia.org/wiki/Amazon_SageMaker (accessed.

[4] "Unified engine for large-scale data analytics." https://spark.apache.org/ (accessed.

[5] "Spark 101: What Is It, What It Does, and Why It Matters." https://developer.hpe.com/blog/spark-101-what-is-it-what-it-does-and-why-it-matters/ (accessed.

[6] "Machine Learning Library (MLlib) Guide." https://spark.apache.org/docs/latest/ml-guide.html (accessed.

[7] "Democratising Machine learning with H2O." https://towardsdatascience.com/democratising-machine-learning-with-h2o-7f2f79e10e3f (accessed.

[8] "Tensorflow." https://www.tensorflow.org/ (accessed.

[9] "Amazon S3." https://aws.amazon.com/s3/ (accessed.

[10] C. Yin, B. Wang, V. J. Gan, M. Wang, and J. C. Cheng, "Automated semantic segmentation of industrial point clouds using ResPointNet++," *Automation in Construction,* vol. 130, p. 103874, 2021.