Learning “Machine Learning” with AWS SageMaker using Titanic dataset

Summarized steps;

Sanitize the data set → upload data to S3 bucket → create SageMaker instance → upload pre-configured Jupiter notebook → train and create the model → use Lambda to invoke endpoint.

Data classification

Data related to Titanic can be downloaded from Kaggle website. It will be in two .csv files (train.csv, test.csv). I’ve combined them together and created “titanic all.xlsx” with 1309 passenger details. Below image is the original data set. With the type of the algorithm data in this format will not work. Therefore, im going to convert below data in to numerical (integer) values.

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerI	Name	Surviv	Pclas	Sex	Age	SibSp	Parch	Ticke	Fare	Cabir	Embark
2	1	Braund, Mr. Owen Harris	0	3	male	22	1	0	A/5 21171	7.25		S
3	2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	1	1	female	38	1	0	PC 17599	71.2833	C85	C
4	3	Heikkinen, Miss. Laina	1	3	female	26	0	0	STON/O2.	7.925		S
5	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	1	female	35	1	0	113803	53.1	C123	S
6	5	Allen, Mr. William Henry	0	3	male	35	0	0	373450	8.05		S
7	6	Moran, Mr. James	0	3	male		0	0	330877	8.4583		Q
8	7	McCarthy, Mr. Timothy J	0	1	male	54	0	0	17463	51.8625	E46	S
9	8	Palsson, Master. Gosta Leonard	0	3	male	2	3	1	349909	21.075		S
10	9	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	3	female	27	0	2	347742	11.1333		S
11	10	Nasser, Mrs. Nicholas (Adele Achem)	1	2	female	14	1	0	237736	30.0708		C
12	11	Sandstrom, Miss. Marguerite Rut	1	3	female	4	1	1	PP 9549	16.7	G6	S
13	12	Bonnell, Miss. Elizabeth	1	1	female	58	0	0	113783	26.55	C103	S
14	13	Saundercock, Mr. William Henry	0	3	male	20	0	0	A/5. 2151	8.05		S
15	14	Andersson, Mr. Anders Johan	0	3	male	39	1	5	347082	31.275		S
16	15	Vestrom, Miss. Hulda Amanda Adolfina	0	3	female	14	0	0	350406	7.8542		S
17	16	Hewlett, Mrs. (Mary D Kingcome)	1	2	female	55	0	0	248706	16		S
18	17	Rice, Master. Eugene	0	3	male	2	4	1	382652	29.125		Q
19	18	Williams, Mr. Charles Eugene	1	2	male		0	0	244373	13		S
20	19	Vander Planke, Mrs. Julius (Emelia Maria Vandemoort	0	3	female	31	1	0	345763	18		S
21	20	Masselmani, Mrs. Fatima	1	3	female		0	0	2649	7.225		C
22	21	Fynney, Mr. Joseph J	0	2	male	35	0	0	239865	26		S

Below image is the legend for the columns in the data set. Some of the columns has to be converted and some has to be dropped.

In a name of a passenger it has the title of the person. Since the name cannot be taken as a whole, I extracted the title (all done in excel). From there I was able to get 7 categories (military, ladies, professions, general 1 – 4). The name column was replaced by the title (social status) column.

Age was categorized by age groups of 10. Some passengers age was missing. For that I checked if the passenger survived or not and accordingly gave an age. Men highest survival age 18 – 30, women highest survival age 14-40.

Categorized the fare by groups of 100 making four groups.

In the cabin number we could get the letter for the deck. Therefore, cabin number is converted to deck number.

index	PassengerId:	NA	Unique Id of a passenger (not used)
result	survival:	2	yes = 1 / no = 0
	Name:	NA	name of the passanger with title (not used)
1	social status	7	7 social groups catagorized by title in the name
2	pclass:	3	Ticket class 1, 2, 3
3	sex:	2	male = 1 / female = 0
4	Age:	8	8 age groups
5	sibsp:	7	# of siblings / spouses aboard the Titanic
6	parch:	8	# of parents / children aboard the Titanic
7	not alone	2	yes = 1 / no = 0
8	fare:	4	4 fare groups
9	cabin:	8	Cabin number. convert to # of decks (1-8)
10	embarked:	3	Port C = 1 / Q = 2 / S = 3
	ticket:	NA	Ticket number (not used)

class	category	social group
1	milatary	capt
1	milatary	col
1	milatary	major
2	ladies	mlle
2	ladies	mme
2	ladies	countess
2	ladies	dona
2	ladies	lady
2	ladies	ms
3	profession	dr
3	profession	rev
3	profession	don
3	profession	jonkheer
3	profession	sir
5	general 1	master
6	general 2	miss
9	general 3	mr
10	general 4	mrs

class	age group
1	0 - 10
2	.11 - 20
3	21 - 30
4	31 - 40
5	41 - 50
6	51 - 60
7	61 - 70
8	70 - 80

class	fare group
1	0 - 100
2	101 - 200
3	201 - 300
4	501 - 600

How title extracted from the name; in excel → select the "name" column → "Data" tab in the ribbon → "Text to Columns" → Delimited → select "space".

	a	b	b1	c	d	e	f	g.	
2	Braund,	Mr.	9	Owen	Harris				
3	Cumings,	Mrs.	10	John	Bradley	(Florence	Briggs	Thayer)	
4	Heikkinen,	Miss.	6	Laina					
5	Futrelle,	Mrs.	10	Jacques	Heath	(Lily	May	Peel)	
6	Allen,	Mr.	9	William	Henry				
7	Moran,	Mr.	9	James					
8	McCarthy,	Mr.	9	Timothy	J				
9	Palsson,	Master.	5	Gosta	Leonard				
10	Johnson,	Mrs.	10	Oscar	W	(Elisabeth	Vilhelmin	Berg)	
11	Nasser,	Mrs.	10	Nicholas	(Adele	Achem)			
12	Sandstrom,	Miss.	6	Marguerite	Rut				
13	Bonnell,	Miss.	6	Elizabeth					
14	Saundercock,	Mr.	9	William	Henry				
15	Andersson,	Mr.	9	Anders	Johan				
16	Vestrom,	Miss.	6	Hulda	Amanda	Adolfina			
17	Hewlett,	Mrs.	10	(Mary	D	Kingcome)			
18	Rice,	Master.	5	Eugene					
19	Williams,	Mr.	9	Charles	Eugene				
20	Vander	Mrs.	10	Planke,	Julius	(Emelia	Maria	Vandemoortele)	
21	Masselmani	Mrs	10	Fatima					

After doing all modifications the data set will look like this. I have selected the purple highlighted columns as the final output.

Passengerld	Name	Survive	social statu	social status (numb	Pclas	Sex	sex (binar	Age	age cla	SibSp	Parch
1	Braund, Mr. Owen Harris	0	Mr.	9	3	male	1	22	3	1	0
2	Cumings, Mrs. John Bradley (Florence Briggs Th	1	Mrs.	10	1	female	0	38	4	1	0
3	Heikkinen, Miss. Laina	1	Miss.	6	3	female	0	26	3	0	0
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	Mrs.	10	1	female	0	35	4	1	0
5	Allen, Mr. William Henry	0	Mr.	9	3	male	1	35	4	0	0
6	Moran, Mr. James	0	Mr.	9	3	male	1	1	1	0	0
7	McCarthy, Mr. Timothy J	0	Mr.	9	1	male	1	54	6	0	0
8	Palsson, Master. Gosta Leonard	0	Master.	5	3	male	1	2	1	3	1
9	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Be	1	Mrs.	10	3	female	0	27	3	0	2
10	Nasser, Mrs. Nicholas (Adele Achem)	1	Mrs.	10	2	female	0	14	2	1	0
11	Sandstrom, Miss. Marguerite Rut	1	Miss.	6	3	female	0	4	1	1	1
12	Bonnell, Miss. Elizabeth	1	Miss.	6	1	female	0	58	6	0	0
13	Saundercock, Mr. William Henry	0	Mr.	9	3	male	1	20	2	0	0
14	Andersson, Mr. Anders Johan	0	Mr.	9	3	male	1	39	4	1	5
15	Vestrom, Miss. Hulda Amanda Adolfina	0	Miss.	6	3	female	0	14	2	0	0
16	Hewlett, Mrs. (Mary D Kingcome)	1	Mrs.	10	2	female	0	55	6	0	0
17	Rice, Master. Eugene	0	Master.	5	3	male	1	2	1	4	1
18	Williams, Mr. Charles Eugene	1	Mr.	9	2	male	1	18	2	0	0
19	Vander Planke, Mrs. Julius (Emelia Maria Vande	0	Mrs.	10	3	female	0	31	4	1	0
20	Masselmani, Mrs. Fatima	1	Mrs.	10	3	female	0	14	2	0	0

Passengerld	not alo	Ticket	Fare	Fare (ir	fare cla	Cabin		deck (letter)	deck (numb	Embarked	Embarked (numbe
1	1	A/5 21171	7.25	7	1			a	1	S	3
2	1	PC 17599	71.2833	71	1	C85	c	c	3	C	1
3	0	STON/O2. 3101	7.925	8	1			b	2	S	3
4	1	113803	53.1	53	1	C123	c	c	3	S	3
5	0	373450	8.05	8	1			c	3	S	3
6	0	330877	8.4583	8	1			d	4	Q	2
7	0	17463	51.8625	52	1	E46	e	e	5	S	3
8	1	349909	21.075	21	1			e	5	S	3
9	1	347742	11.1333	11	1			g	7	S	3
10	1	237736	30.0708	30	1			t	8	C	1
11	1	PP 9549	16.7	17	1	G6	g	g	7	S	3
12	0	113783	26.55	27	1	C103	c	c	3	S	3
13	0	A/5. 2151	8.05	8	1			t	8	S	3
14	1	347082	31.275	31	1			a	1	S	3
15	0	350406	7.8542	8	1			b	2	S	3
16	0	248706	16	16	1			c	3	S	3
17	1	382652	29.125	29	1			d	4	Q	2
18	0	244373	13	13	1			e	5	S	3
19	1	345763	18	18	1			f	6	S	3
20	0	2649	7.225	7	1			g	7	C	1

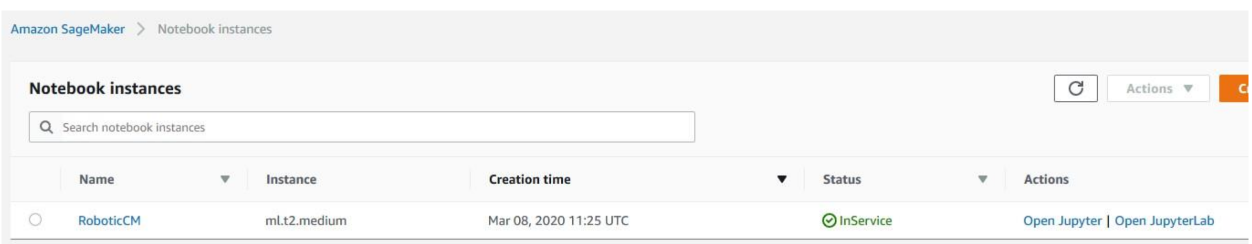
The final output. From this I created the “titanic-all.csv” file. which will be uploaded to S3 bucket for model training.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Index - Passenge rId	Result - Survive	social status (numbe	Pclas	sex (binar	age cla	SibSp	Parch	not alo	fare cla	deck (numbe	Embarked (numbe
2	1	0	9	3	1	3	1	0	1	1	1	3
3	2	1	10	1	0	4	1	0	1	1	3	1
4	3	1	6	3	0	3	0	0	0	1	2	3
5	4	1	10	1	0	4	1	0	1	1	3	3
6	5	0	9	3	1	4	0	0	0	1	3	3
7	6	0	9	3	1	1	0	0	0	1	4	2
8	7	0	9	1	1	6	0	0	0	1	5	3
9	8	0	5	3	1	1	3	1	1	1	5	3
10	9	1	10	3	0	3	0	2	1	1	7	3
11	10	1	10	2	0	2	1	0	1	1	8	1
12	11	1	6	3	0	1	1	1	1	1	7	3
13	12	1	6	1	0	6	0	0	0	1	3	3
14	13	0	9	3	1	2	0	0	0	1	8	3
15	14	0	9	3	1	4	1	5	1	1	1	3
16	15	0	6	3	0	2	0	0	0	1	2	3
17	16	1	10	2	0	6	0	0	0	1	3	3
18	17	0	5	3	1	1	4	1	1	1	4	2
19	18	1	9	2	1	2	0	0	0	1	5	3
20	19	0	10	3	0	4	1	0	1	1	6	3
21	20	1	10	3	0	2	0	0	0	1	7	1

Creating SageMaker instance

First of all, we need to create a S3 bucket to upload data and this will be used to store the output of the training.

Second step is to create SageMaker notebook instance.



Executing the Jupiter Notebook and training

I have slightly modified (data classification part is removed as for my scenario all the classification and conversion is performed earlier) Jupiter notebook. Open Jupiter and upload the notebook.

Make sure to change necessary areas according to your naming conversions

```
# Build, Train, Deploy change management data with AWS
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
```

In [5]:
```
# AWS Specific Imports and Setup

import boto3
from sagemaker import get_execution_role

role = get_execution_role()

region = boto3.Session().region_name

bucket='roboticcm' # Replace with your s3 bucket name
prefix = 'linear-svc' # Used as part of the path in the bucket where you store data
bucket_path = 'https://roboticcm.s3.amazonaws.com/'.format(region,bucket) # The URL to access the bucket

raw_titanic_data = 's3://{}/{}'.format(bucket, 'titanic-all.csv')

print(raw_titanic_data)
```

```
s3://roboticcm/titanic-all.csv
```

In [6]:
```
titanic = pd.read_csv(raw_titanic_data)
```

Let's look over what data we have and a little bit about how it is structured. The 'info' function does a good job at showing what fields have null values, and we can learn about the different data types of our individual values.

In [7]:
```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
Index-PassengerId    1309 non-null int64
Result-Survived      1309 non-null int64
social_status        1309 non-null int64
Pclass               1309 non-null int64
sex                  1309 non-null int64
age_group            1309 non-null int64
SibSp                1309 non-null int64
Parch                1309 non-null int64
not_alone            1309 non-null int64
fare_group           1309 non-null int64
deck                 1309 non-null int64
Embarked             1309 non-null int64
dtypes: int64(12)
memory usage: 122.8 KB
```

The head function also allows us to see the first 'n' amount of rows. This is great for diving a little deeper into what our dataset contains.

In [8]:
```
titanic.head(10)
```

Out[8]:

	Index-PassengerId	Result-Survived	social_status	Pclass	sex	age_group	SibSp	Parch	not_alone	fare_group	deck	Embarked
0	1	0	9	3	1	3	1	0	1	1	1	3
1	2	1	10	1	0	4	1	0	1	1	3	1
2	3	1	6	3	0	3	0	0	0	1	2	3
3	4	1	10	1	0	4	1	0	1	1	3	3
4	5	0	9	3	1	4	0	0	0	1	3	3
5	6	0	9	3	1	1	0	0	0	1	4	2
6	7	0	9	1	1	6	0	0	0	1	5	3
7	8	0	5	3	1	1	3	1	1	1	5	3
8	9	1	10	3	0	3	0	2	1	1	7	3
9	10	1	10	2	0	2	1	0	1	1	8	1

Let's clean up our dataset. For our quick analysis, let's remove the columns or features that had a low correlation with our survived column. Let's also remove a few other features that we aren't going to try to parse to derive additional value. BUT! You absolutely could or would in a real situation. We just aren't going to for the nature of our quick demo!

In [9]:
```
titanic = titanic.drop('Index-PassengerId', 1)
#titanic = titanic.drop('Ticket', 1)
#titanic = titanic.drop('Cabin', 1)
#titanic = titanic.drop('Embarked', 1)
#titanic = titanic.drop('Age', 1)
#titanic = titanic.drop('SibSp', 1)
#titanic = titanic.drop('Parch', 1)
```

Now if we look at our data, we have a much more simple data set.

In [10]:
```
titanic.head()
```

Out[10]:

	Result-Survived	social_status	Pclass	sex	age_group	SibSp	Parch	not_alone	fare_group	deck	Embarked
0	0	9	3	1	3	1	0	1	1	1	3
1	1	10	1	0	4	1	0	1	1	3	1
2	1	6	3	0	3	0	0	0	1	2	3
3	1	10	1	0	4	1	0	1	1	3	3
4	0	9	3	1	4	0	0	0	1	3	3

In [11]:
```python
titanic.describe()
```

Out[11]:

	Result-Survived	social_status	Pclass	sex	age_group	SibSp	Parch	not_alone	fare_group	deck	Embarked
count	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000	1309.000000
mean	0.377387	8.195569	2.294882	0.644003	3.669977	0.498854	0.385027	0.396486	1.096257	4.240642	2.491979
std	0.484918	1.789780	0.837836	0.478997	1.668064	1.041658	0.865560	0.489354	0.396729	2.183690	0.814230
min	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000
25%	0.000000	6.000000	2.000000	0.000000	3.000000	0.000000	0.000000	0.000000	1.000000	2.000000	2.000000
50%	0.000000	9.000000	3.000000	1.000000	3.000000	0.000000	0.000000	0.000000	1.000000	4.000000	3.000000
75%	1.000000	9.000000	3.000000	1.000000	5.000000	1.000000	0.000000	1.000000	1.000000	6.000000	3.000000
max	1.000000	10.000000	3.000000	1.000000	8.000000	8.000000	9.000000	1.000000	4.000000	8.000000	3.000000

Now that we have our data cleaned and ready, we are going to split our data into a 2/3, 1/3 split of training vs testing.

In [14]:
```python
features = titanic.drop('Result-Survived', 1)
labels = titanic['Result-Survived']

train, test, train_labels, test_labels = train_test_split(features,
                                                          labels,
                                                          test_size=0.33, random_state=42)
```

Sagemaker needs the data to be in S3, so we are going to now need to move our split datasets into S3 so that we can do further analysis.

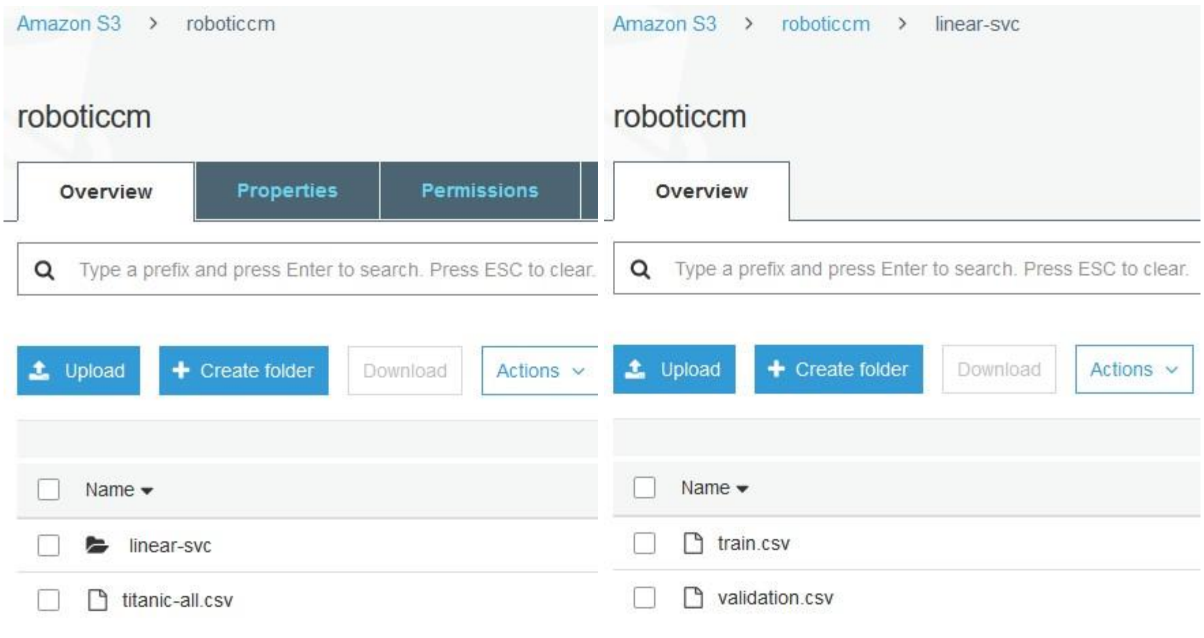
In [15]:
```python
from io import StringIO

test_csv_buffer = StringIO()
train_csv_buffer = StringIO()
pd.concat([test_labels, test], axis=1).to_csv(test_csv_buffer, header=True, index=False)
pd.concat([train_labels, train], axis=1).to_csv(train_csv_buffer, header=True, index=False)

s3_resource = boto3.resource('s3')
s3_resource.Object(bucket, prefix + '/train.csv').put(Body=train_csv_buffer.getvalue())
s3_resource.Object(bucket, prefix + '/validation.csv').put(Body=test_csv_buffer.getvalue())
```

Out[15]:

```
{'ResponseMetadata': {'RequestId': '02F4EAC964B0B45C',
  'HostId': '3qig2ZjGYRvMXS96jzNGvWXuDfAsdiX1IUHnPfJy7408wFKkArXW558B0WahYB2hieTY9WsomIM=',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': '3qig2ZjGYRvMXS96jzNGvWXuDfAsdiX1IUHnPfJy7408wFKkArXW558B0WahYB2hieTY9WsomIM=',
   'x-amz-request-id': '02F4EAC964B0B45C',
   'date': 'Sun, 08 Mar 2020 12:13:37 GMT',
   'etag': '"3600f49fa07e19f35cf411a303e4442c"',
   'content-length': '0',
   'server': 'AmazonS3'},
  'RetryAttempts': 0},
 'ETag': '"3600f49fa07e19f35cf411a303e4442c"'}
```



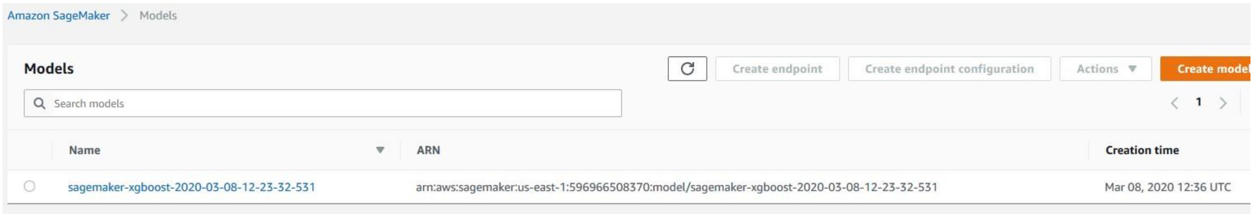
Once you are ready, we can train the model with the 'fit' method. The actual training time can vary, but this is what is actually building out your model and model artifacts.

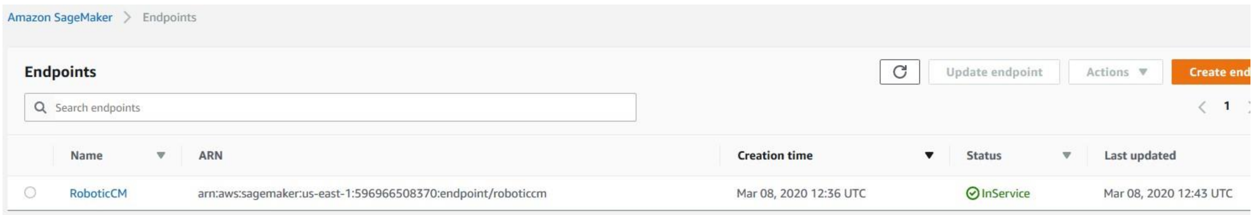
In [25]:
```
xgb_model.fit(inputs=data_channels,  logs=True)
```

```
2020-03-08 12:23:32 Starting - Starting the training job...
2020-03-08 12:23:34 Starting - Launching requested ML instances......
2020-03-08 12:24:39 Starting - Preparing the instances for training......
2020-03-08 12:25:37 Downloading - Downloading input data...
2020-03-08 12:26:32 Training - Training image download completed. Training in progress.
2020-03-08 12:26:32 Uploading - Uploading generated training model.INFO:sagemaker-containers:Imported framework sagemaker_x
gboost_container.training
INFO:sagemaker-containers:Failed to parse hyperparameter objective value multi:softmax to Json.
Returning the value itself
INFO:sagemaker-containers:No GPUs detected (normal if no gpus installed)
INFO:sagemaker_xgboost_container.training:Running XGBoost Sagemaker in algorithm mode
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
INFO:root:Determined delimiter of CSV input is ','
[12:26:29] 878x10 matrix with 8780 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
INFO:root:Determined delimiter of CSV input is ','
[12:26:29] 433x10 matrix with 4330 entries loaded from /opt/ml/input/data/validation?format=csv&label_column=0&delimiter=,
INFO:root:Single node training.
INFO:root:Train matrix has 878 rows
INFO:root:Validation matrix has 433 rows
[0]#011train-merror:0.133257#011validation-merror:0.124711
[1]#011train-merror:0.133257#011validation-merror:0.124711
[2]#011train-merror:0.133257#011validation-merror:0.124711
[3]#011train-merror:0.133257#011validation-merror:0.124711
[4]#011train-merror:0.133257#011validation-merror:0.124711
[5]#011train-merror:0.133257#011validation-merror:0.124711
[6]#011train-merror:0.133257#011validation-merror:0.124711
[7]#011train-merror:0.133257#011validation-merror:0.124711
[8]#011train-merror:0.133257#011validation-merror:0.124711
[9]#011train-merror:0.133257#011validation-merror:0.124711

2020-03-08 12:26:39 Completed - Training job completed
Training seconds: 62
Billable seconds: 62
```

After training complete we can see the model and endpoint.





Using Lambda function to call predictions from the model

In Lambda create a Node.js function. Create an IAM with "sagemaker:InvokeEndpoint" or full access.

After creating the function copy below code to the function. Create "environment variable" with the endpoint name.

- -------------------------------

```
var AWS = require('aws-sdk');

exports.handler = (event) => {
    const sagemakerruntime = new AWS.SageMakerRuntime({region: 'us-west-2'});
    let csv = Object.values(event).join(",");

    const params = {
        Body: csv,
        EndpointName: process.env.ENDPOINT_NAME,
        ContentType: 'text/csv',
        Accept: 'text/csv'
    };

    return new Promise((resolve, reject) => {
        return sagemakerruntime.invokeEndpoint(params, (err, data) => {
            if (err) {
                let response = {
                    statusCode: 400,
                    body: JSON.stringify(err),
                };
                reject(response);
            }
            else {
```

```
                let willSurviveResponse = 'Yes, You will Survive the
Titanic!';
                let survived = 'Yes'
                if (JSON.parse(Buffer.from(data.Body).toString('utf8')) ===
0) {
                    survived = 'No'
                    willSurviveResponse = 'No... Perhaps you should avoid
boats.';
                }

                let response = {
                    statusCode: 200,
                    survived: survived,
                    message: willSurviveResponse,
                };
                resolve(response);
            }
        });
    });
};
```

- -------------------------------

Create test case with below format.

- -------------------------------

{

"social_status": "10",

"Pclass": "2",

"sex": "0",

"age_group": "2",

"SibSp": "1",

"Parch": "0",

"not_alone": "1",

"fare_group": "1",

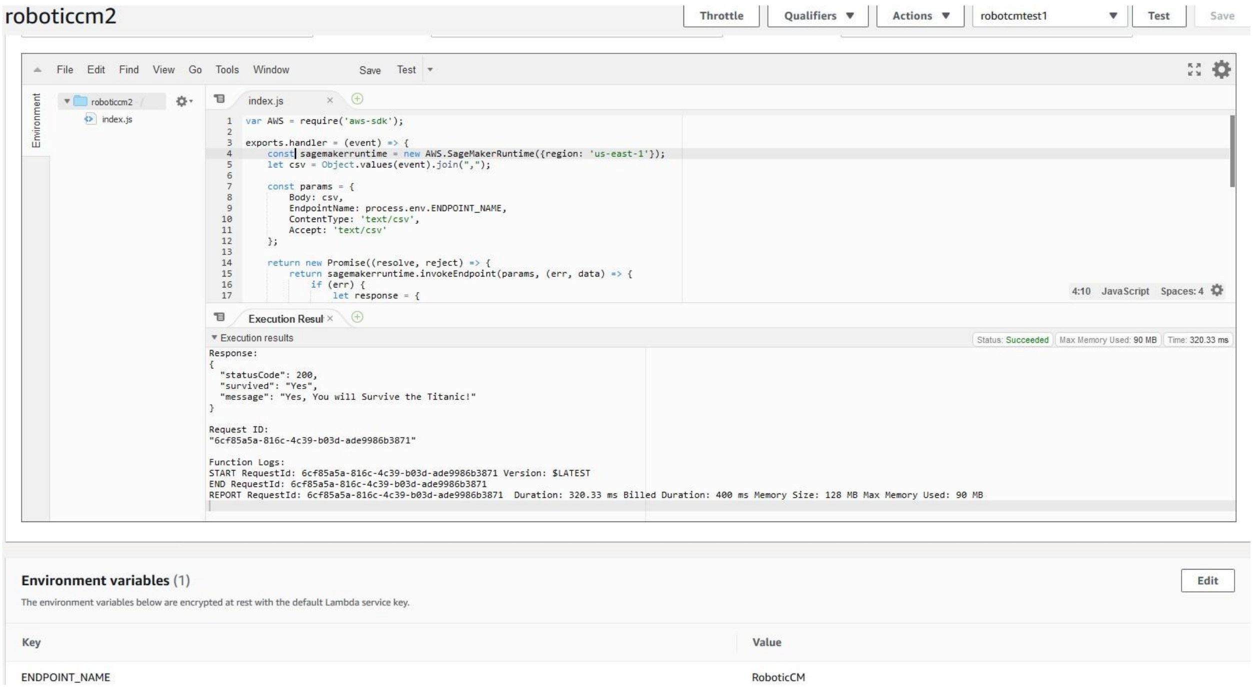
"deck": "8",

"Embarked": "1"

}

- -------------------------------

Test cases will provide results as below.



Reference:

https://www.kaggle.com/c/titanic/data

(good for data classification)

https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8

https://drakeloud.com/blog/ai-ml-go-from-jupyter-notebook-to-deployed-endpoints-with-sagemaker/

https://aws.amazon.com/blogs/machine-learning/call-an-amazon-sagemaker-model-endpoint-using-amazon-api-gateway-and-aws-lambda/