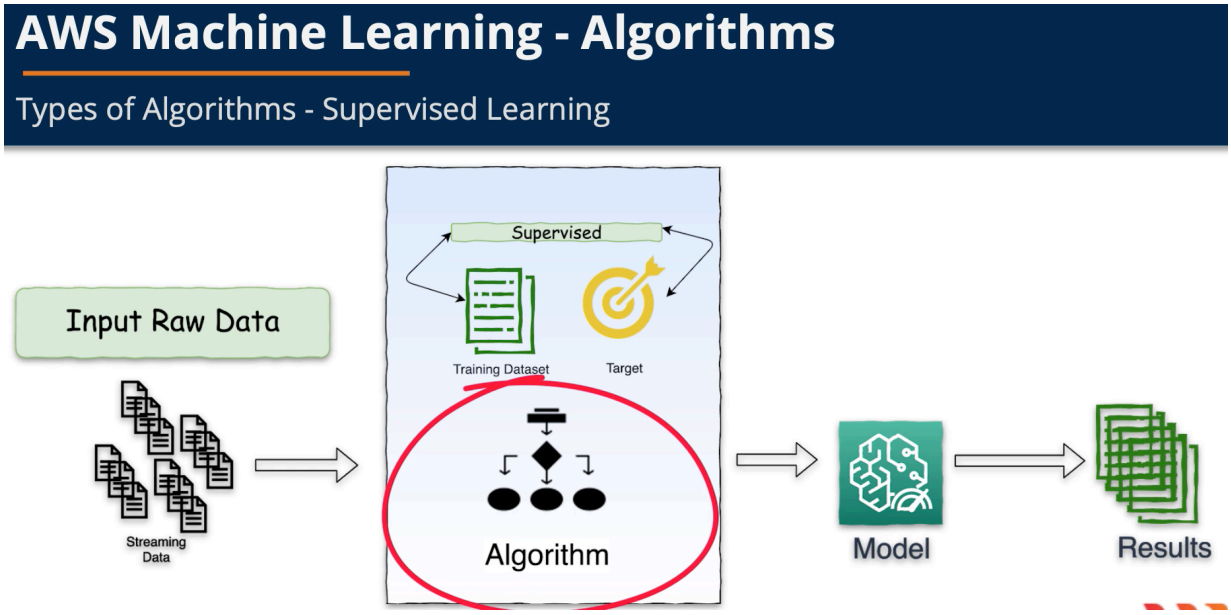
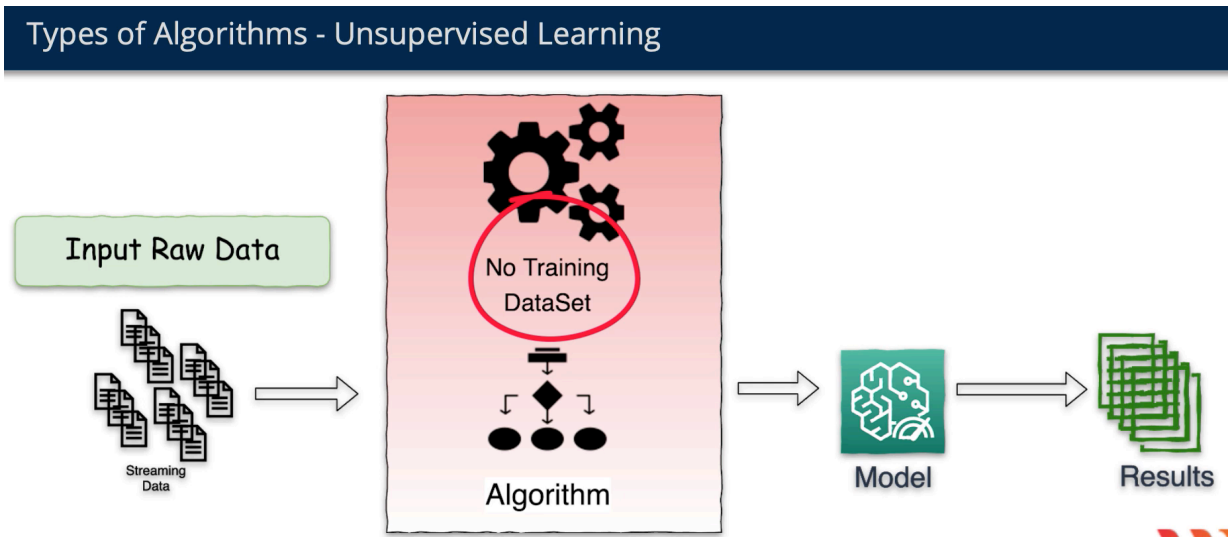
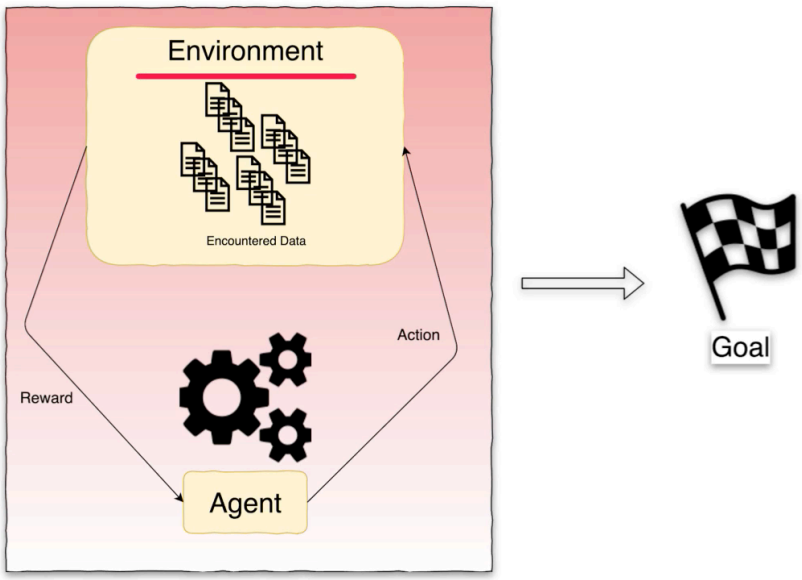
Whizlabs - ML Specialty Exam Course - Algorithms - part 1

https://www.whizlabs.com/learn/course/aws-mls-practice-tests/video/3474





Types of Algorithms - Reinforcement Learning



SageMaker - Algorithms for Any Kind of Data

AWS Machine Learning - Algorithms

Algorithms for Any Kind of Data

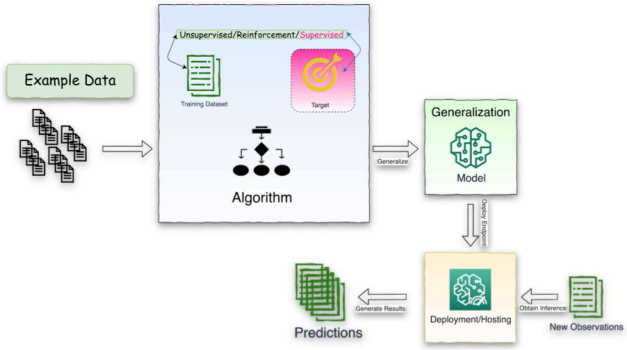
- ❑ Structured data
  - ❑ Linear Learner
  - ❑ Factorization Machines
  - ❑ XGBoost
  - ❑ K-Means
  - ❑ Principal Component Analysis
  - ❑ Random Cut Forest
- ❑ Image data
  - ❑ Image Classification
- ❑ Natural Language data
  - ❑ Sequence2Sequence
  - ❑ Neural Topic Modeling
  - ❑ Latent Dirichlet Allocation
  - ❑ Blazing Text
- ❑ Time Series data
  - ❑ DeepAR

SageMaker Built-in Algorithms

1. Regression algorithms
2. Clustering algorithms
3. Classification algorithms
4. Image analysis algorithms
5. Text analysis algorithms
6. Anomaly detection algorithms
7. Reinforcement algorithms
8. Forecasting algorithms

Machine Learning Algorithms - Generalization

- Algorithms use example data to create a generalization (a *model*) that answers your business question
- After creating a model using example data, use it to answer your business question with new data (obtaining an *inference*)



Machine Learning Algorithms - Understand the Business Problem

- The type of answer sought influences your choice of algorithm - Discrete Categories
  - Direct mail campaign to solicit political donations
  - Discrete category type answers
    - "Considering past solicitation responses, mail to this contributor?"
      - Binary: yes/no
    - "Considering past solicitor segmentation, which segment is the contributor in?"
      - Multi-class: "small donation", "significant donation", "corporate donor"
  - SageMaker has built-in algorithms for discrete classification problems like these
    - Linear Learner: set `predictor_type` hyperparameter to `binary_classifier`
    - XGBoost: set `objective` hyperparameter to `reg:logistic`

We could also use multiclass_classifier for LinearLearner:

`predictor_type`	Specifies the type of target variable as a binary classification, multiclass classification, or regression. **Required** Valid values: `binary_classifier`, `multiclass_classifier`, or `regressor`

XGboost: https://github.com/dmlc/xgboost/blob/master/doc/parameter.rst#learning-task-parameters

`objective`	Specifies the learning task and the corresponding learning objective. Examples: `reg:logistic`, `multi:softmax`, `reg:squarederror`. For a full list of valid inputs, refer to XGBoost Learning Task Parameters. **Optional** Valid values: string Default value: `reg:squarederror`

Machine Learning Algorithms - Understand the Business Problem

- The type of answer sought influences your choice of algorithm - Quantitative
  - Direct mail campaign to solicit political donations
  - Quantitative type answers
    - "Considering the ROI on past solicitation mailings, what is the ROI for soliciting this donor?"
      - Quantitative: higher ROI donors get mailing
  - SageMaker has built-in algorithms for quantitative analysis problems like these
    - Linear Learner: set `predictor_type` hyperparameter to `regressor`
    - XGBoost: set `objective` hyperparameter to `reg:linear`

Machine Learning Algorithms - Understand the Business Problem

- The type of answer sought influences your choice of algorithm - Discrete Recommendations
  - Direct mail campaign to solicit political donations
  - Discrete recommendation type answers
    - “Considering past solicitation mailing responses, what is the recommended content for each donor?”
  - SageMaker has built-in algorithms for discrete recommendation problems like these
    - Factorization Machines for recommendations

Machine Learning Algorithms - Understand the Business Problem

- The type of answer sought influences your choice of algorithm - Identifying Groups
  - Direct mail campaign to solicit political donations
  - Identifying Group type answers
    - “Group potential and current donors into 12 groups based on their attributes, how should they be grouped?”
    - Send mailing to donors in the group that has the highest percentage of current donors, i.e. potential donors that are most similar to current donors
  - SageMaker has built-in algorithms for group identification problems like these
    - K-Means

Machine Learning Algorithms - Understand the Business Problem

- The type of answer sought influences your choice of algorithm - Dimensionality Reduction
  - Direct mail campaign to solicit political donations
  - Dimensionality type answers
    - ”What attributes differentiate donors, what are the relative values for the donors along these dimensions?”
    - Use to simplify the view of current and prospective donors and gain clarity on value of donor attributes
  - SageMaker has built-in algorithms for dimensionality problems like these
    - Principal Component Analysis

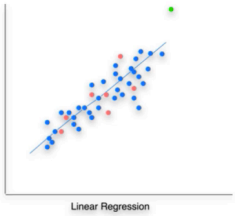
Machine Learning Algorithms - Understand the Business Problem

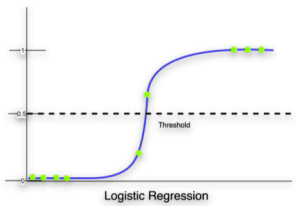
- Other algorithms for specific use cases
  - Classifying images
    - Image Classification
  - Translation
    - Sequence-to-Sequence
  - Determining topics of sets of documents
    - Latent Dirichlet Allocation
    - Neural Topic Model
  - Text classification
    - Blazing Text
  - Anomaly detection
    - Random Cut Forest
    - k-Nearest-Neighbors

Regression Algorithms

Regression Algorithm - Defined

- Supervised learning algorithm
- Performs a regression task where it models a target (dependent variable) prediction based on a vector of independent variables
- For linear regression the goal is to find the best-fit regression line through the independent variable(s) as related to the dependent variable
- Minimize error between predicted value and observed value in the training data





Example Regression Algorithm Use Cases

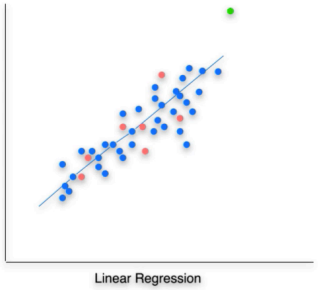
- Optimizing pricing for product line
- Predicting whether a customer will default on a loan
- Predicting whether a patient has cancer based on image scan data
- Predicting user churn
- Sales forecasting
- Predict whether a voter will select a candidate or not
- Predict house prices

LinearLearner Algorithm for Regression

AWS Machine Learning - Regression Algorithms

SageMaker Regression Algorithms - Linear Learner

- Linear Learner
  - Input a set of high-dimensional vectors including a numeric target, or label
  - Target is a real number
  - Learns a linear function and maps a vector to an approximation of the target
  - Good model based on Linear Learner optimizes
    - Continuous objective: mean square error, cross entropy loss, absolute error
  - Requires a data matrix of observations across dimension of features
  - Also requires a target column across the observations
  - Example use case: predict house prices based on housing data including location, square feet, etc.



XGboost for Regression

AWS Machine Learning - Regression Algorithms

SageMaker Regression Algorithms - XGBoost

- XGBoost
  - Implementation of gradient boosted trees algorithm
  - Supervised learning algorithm for predicting a target by combining the estimates from a set of simpler models
  - Requires a data matrix of observations across dimension of features
  - Also requires a target column across the observations
  - Can differentiate the importance of features through weights
  - Example use case: predict income based on census data

K-Nearest Neighbors for Regression

AWS Machine Learning - Regression Algorithms

SageMaker Regression Algorithms - K-Nearest Neighbors

- K-Nearest Neighbors
  - Finds the k closest points to the sample point and gives a prediction of the average of their features
  - Indexed based
  - Objective: build k-NN index to allow for efficient determination of the distance between points
  - Train to construct the index
  - Use dimensionality reduction to avoid the “curse of dimensionality”
  - Example use case: predict student absenteeism using student grades, demographic, social, and school related features

Factorization Machines for Regression

AWS Machine Learning - Regression Algorithms

SageMaker Regression Algorithms - Factorization Machines

- Factorization Machines
  - Extension of linear model used on high dimensional sparse datasets
  - Typically used for sparse datasets such as click prediction and item recommendation
  - Continuous objective: Root Mean Square Error
  - Example use case: analyze the images of handwritten digits

Important hyperparameters for Regression Algorithms

SageMaker Regression Algorithms - Important Hyperparameters

- Linear Learner
  - feature_dim: number of feature in the input
  - predictor_type: type of the target variable (regressor for regression problems)
  - loss: specifies the loss function (auto, squared_loss, absolute_loss, etc.)
- XGBoost
  - num_round: number of rounds the training runs
  - objective: learning task and learning objective (i.e. reg:logistic, reg:squarederror)

SageMaker Regression Algorithms - Important Hyperparameters

- K-Nearest Neighbors
  - feature_dim: number of feature in the input
  - k: number of nearest neighbors
  - predictor_type: regressor for regression problems
  - sample_size: number of data points to be samples from the training dataset
  - dimensionality_reductoin_target: target dimension to reduce to
- Factorization Machines
  - feature_dim: number of feature in the input
  - num_factors: dimensionality of factorization
  - predictor_type: regressor for regression problems

Regression algorithms - Lab

We are going to build a linear learner regression based algorithm, using UCI bike share dataset

Bike Sharing Dataset Data Set

Download: Data Folder, Data Set Description

Abstract: This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information.

Data Set Characteristics:	Univariate	**Number of Instances:**	17389	**Area:**	Social
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	16	**Date Donated**	2013-12-20
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	447044

In [1]:
```
import numpy as np
import boto3
import sagemaker
import io
import sagemaker.amazon.common as smac
import os
import pandas as pd

# Read csv from s3.
# Download from your S3 bucket the bike share data CSV file based on the publically available bike share data from the i
from io import StringIO
s3 = boto3.resource('s3')
bucket = 'machine-learning-exam' # place the day.csv file in a bucket in your account
object_key = 'day.csv'

# Load the data into a pandas dataframe
csv_obj = s3.Object(bucket, object_key)
csv_string = csv_obj.get()['Body'].read().decode('utf-8')

dataset = pd.read_csv(StringIO(csv_string))
dataset.head()
```

Out[1]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

LinearLearner does not like categorical fields so we will replace it

In [2]:
```
# Convert categorical date field
dataset['dteday'] = dataset['dteday'].str.replace("-","")
dataset.head()
```

Out[2]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	20110101	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	20110102	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	20110103	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	20110104	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	20110105	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Randomize and split data 70/30

In [3]:
```
# Randomize the data and split it between train and test datasets on a 70% 30% split respectively
train_data, test_data = np.split(dataset.sample(frac=1, random_state=1729), [int(0.7 * len(dataset))])
print(train_data.shape, test_data.shape)
```

```
(511, 16) (220, 16)
```

511 in train, 220 in test / total of 15 features. 16th column is our label/target

Grab the 15 features and separately the ‘cnt’ feature that is our target
Store then as float32

In [4]:
```
# Get the features and labels.
feature_dataset = train_data[['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
                              'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered' ]]
features = np.array(feature_dataset.values).astype('float32')

label_dataset= train_data[['cnt']]
labels = np.array(label_dataset.values).astype('float32')
labels_vec = np.squeeze(np.asarray(labels))
```

LinearLearner works best with protobuf IO

In [5]:
```
# Setup protoBuf
buffer = io.BytesIO()
smac.write_numpy_to_dense_tensor(buffer, features, labels_vec)
buffer.seek(0)

prefix = 'realestate'
key = 'linearregression'
boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train', key)).upload_fileobj(buffer)
s3_training_data_location = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
print('training dataset will be uploaded to: {}'.format(s3_training_data_location))
```

```
training dataset will be uploaded to: s3://machine-learning-exam/realestate/train/linearregression
```

Set output s3 bucket

In [6]:
```
output_location = 's3://{}/{}/output'.format(bucket, prefix)
print('model artifacts will be uploaded to: {}'.format(output_location))
```

```
model artifacts will be uploaded to: s3://machine-learning-exam/realestate/output
```

Get LinearLearner container image from SageMaker

In [7]:
```
# Get the Linear Learner container instance
from sagemaker.amazon.amazon_estimator import get_image_uri
linear_container = get_image_uri(boto3.Session().region_name, 'linear-learner')
```

Now train the dataset
We pass the hyper parameters with predictor_type = ‘regressor’ and number of features: 15

In [8]:
```
# Train the model
from sagemaker import get_execution_role

role = get_execution_role()

sagemaker_session = sagemaker.Session()

# Provide the container, role, instance type and model output location
linear = sagemaker.estimator.Estimator(linear_container,
                                       role=role,
                                       train_instance_count=1,
                                       train_instance_type='ml.c4.xlarge',
                                       output_path=output_location,
                                       sagemaker_session=sagemaker_session)

# Provide the number of features identified during data preparation
# Provide the predictor_type

linear.set_hyperparameters(feature_dim=15,
                           mini_batch_size=4,
                           predictor_type='regressor')

# Train the model using the previously prepared test data and validate the
# data by providing the validation data.

linear.fit({'train': s3_training_data_location})
```

```
2020-02-14 19:06:55 Starting - Starting the training job...
2020-02-14 19:06:56 Starting - Launching requested ML instances......
2020-02-14 19:08:02 Starting - Preparing the instances for training......
2020-02-14 19:09:24 Downloading - Downloading input data...
2020-02-14 19:09:47 Training - Downloading the training image..Docker entrypoint called with argument(s): train
[02/14/2020 19:10:09 INFO 140163623442240] Reading default configuration from /opt/amazon/lib/python2.7/site-package
s/algorithm/resources/default-input.json: {u'loss_insensitivity': u'0.01', u'epochs': u'15', u'feature_dim': u'auto',
u'init_bias': u'0.0', u'lr_scheduler_factor': u'auto', u'num_calibration_samples': u'10000000', u'accuracy_top_k':
u'3', u'_num_kv_servers': u'auto', u'use_bias': u'true', u'num_point_for_scaler': u'10000', u'_log_level': u'info',
u'quantile': u'0.5', u'bias_lr_mult': u'auto', u'lr_scheduler_step': u'auto', u'init_method': u'uniform', u'init_sigm
a': u'0.01', u'lr_scheduler_minimum_lr': u'auto', u'target_recall': u'0.8', u'num_models': u'auto', u'early_stopping_
patience': u'3', u'momentum': u'auto', u'unbias_label': u'auto', u'wd': u'auto', u'optimizer': u'auto', u'_tuning_obj
```

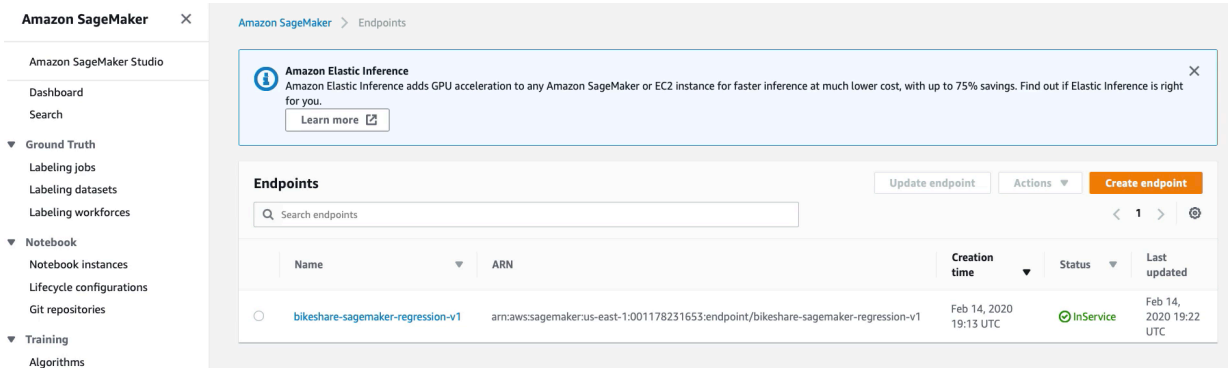
```
[02/14/2020 19:10:47 INFO 140163623442240] #quality_metric: host=algo-1, train absolute_loss <loss>=1.04397015656
[02/14/2020 19:10:47 INFO 140163623442240] Best model found for hyperparameters: {"lr_scheduler_step": 100, "wd": 0.0
001, "optimizer": "adam", "lr_scheduler_factor": 0.99, "l1": 0.0, "learning_rate": 0.005, "lr_scheduler_minimum_lr":
0.0001}
[02/14/2020 19:10:47 INFO 140163623442240] Saved checkpoint to "/tmp/tmpRVYAEU/mx-mod-0000.params"
[02/14/2020 19:10:47 INFO 140163623442240] Test data is not provided.
#metrics {"Metrics": {"totaltime": {"count": 1, "max": 37502.739906311035, "sum": 37502.739906311035, "min": 37502.73
9906311035}, "finalize.time": {"count": 1, "max": 145.23792266845703, "sum": 145.23792266845703, "min": 145.237922668
45703}, "initialize.time": {"count": 1, "max": 421.84996604919434, "sum": 421.84996604919434, "min": 421.849966049194
34}, "check_early_stopping.time": {"count": 15, "max": 1.1441707611083984, "sum": 15.458822250366211, "min": 0.907897
94921875}, "setuptime": {"count": 1, "max": 26.974916458129883, "sum": 26.974916458129883, "min": 26.97491645812988
3}, "update.time": {"count": 15, "max": 2790.6999588012695, "sum": 36829.829931259155, "min": 2144.9930667877197}, "e
pochs": {"count": 1, "max": 15, "sum": 15.0, "min": 15}}, "EndTime": 1581707447.206031, "Dimensions": {"Host": "algo-
1", "Operation": "training", "Algorithm": "Linear Learner"}, "StartTime": 1581707409.791645}


Training seconds: 90
Billable seconds: 90
```

Now deploy the model

In [ ]:
```
# Deploy the model
linear_predictor = linear.deploy(initial_instance_count=1,
                                 instance_type='ml.c4.xlarge',
                                 endpoint_name='bikeshare-sagemaker-regression-v1')
```

```
------
```

We can see it in SageMaker Console



Now get predictions, looping through the test data
We print out the prediction as well as the the actual

In [10]:
```
# Get prediction using the test data
from sagemaker.predictor import csv_serializer, json_deserializer

linear_predictor.content_type = 'text/csv'
linear_predictor.serializer = csv_serializer
linear_predictor.deserializer = json_deserializer

test_feature_dataset = test_data[['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
                                  'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered']]

test_actuals = np.array(test_data['cnt'].astype('float32'))
test_features = np.array(test_feature_dataset.values).astype('float32')

predictions = []
actuals = []
for tf, actual in zip(test_features, test_actuals):
    prediction = linear_predictor.predict(tf)
    predictions.append(prediction['predictions'][0]['score'])
    actuals.append(actual)
    print('prediction: ', prediction['predictions'][0]['score'], '\t\tactual: ', str(actual))
```

```
prediction:  799.390625                actual:  801.0
prediction:  5216.703125               actual:  5217.0
prediction:  7765.5625         actual:  7767.0
prediction:  6850.9375         actual:  6852.0
prediction:  2209.953125               actual:  2209.0
prediction:  6289.296875               actual:  6290.0
prediction:  4790.71875                actual:  4792.0
prediction:  1866.8125         actual:  1865.0
prediction:  5670.359375               actual:  5668.0
prediction:  4492.265625               actual:  4492.0
prediction:  4368.203125               actual:  4367.0
prediction:  2402.453125               actual:  2402.0
prediction:  3847.546875               actual:  3846.0
prediction:  4787.96875                actual:  4788.0
prediction:  3189.03125                actual:  3190.0
prediction:  3006.515625               actual:  3005.0
prediction:  7284.65625                actual:  7286.0
prediction:  2131.25           actual:  2132.0
prediction:  5462.21875                actual:  5464.0
```

Calculate accuracy across all, using cosine similarity:

In [11]:
```
# Get accuracy using Cosine Similarity method
from numpy import dot
from numpy.linalg import norm
tolerance = 1e-10
accuracy = (dot(actuals, predictions)/(norm(actuals)*norm(predictions))) * 100
print('accuracy: ', accuracy)
```

```
accuracy:  100.00000313584616
```

=> spot on with accuracy - it cannot be above 100%

```
In [12]: # delete the endpoint
         sagemaker.Session().delete_endpoint(linear_predictor.endpoint)
```

Now let’s use a client to try out the endpoint

client.py > ...

```python
import os
import io
import boto3
import json
import csv

ENDPOINT_NAME = 'bikeshare-sagemaker-regression-v1'
runtime= boto3.client('runtime.sagemaker')

# Convert categorical date field
observation = "3,2011-01-03,1,0,1,0,1,1,1,0.196364,0.189405,0.437273,0.248309,120,1229"
observation = observation.replace("-","")

data = {"data":observation}
actual = {"data":"1349"}
payload = data['data']
# print(payload)
```

```python

response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                   ContentType='text/csv',
                                   Body=payload)
# print(response)
```

```python

result = json.loads(response['Body'].read().decode())
print('prediction: ', result['predictions'][0]['score'], '\t\tactual: ', str(actual['data']))

```

Bike share csv file:

```
instant,dteday,season,yr,mnth,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,casual,registered,cnt
1,2011-01-01,1,0,1,0,6,0,2,0.344167,0.363625,0.805833,0.160446,331,654,985
2,2011-01-02,1,0,1,0,0,0,2,0.363478,0.353739,0.696087,0.248539,131,670,801
3,2011-01-03,1,0,1,0,1,1,1,0.196364,0.189405,0.437273,0.248309,120,1229,1349
4,2011-01-04,1,0,1,0,2,1,1,0.2,0.212122,0.590435,0.160296,108,1454,1562
5,2011-01-05,1,0,1,0,3,1,1,0.226957,0.22927,0.436957,0.1869,82,1518,1600
6,2011-01-06,1,0,1,0,4,1,1,0.204348,0.233209,0.518261,0.0895652,88,1518,1606
7,2011-01-07,1,0,1,0,5,1,2,0.196522,0.208839,0.498696,0.168726,148,1362,1510
8,2011-01-08,1,0,1,0,6,0,2,0.165,0.162254,0.535833,0.266804,68,891,959
9,2011-01-09,1,0,1,0,0,0,1,0.138333,0.116175,0.434167,0.36195,54,768,822
10,2011-01-10,1,0,1,0,1,1,1,0.150833,0.150888,0.482917,0.223267,41,1280,1321
11,2011-01-11,1,0,1,0,2,1,2,0.169091,0.191464,0.686364,0.122132,43,1220,1263
```

```
Office-Mac-mini:linearLearnerClient vincentbloise$ python client.py
prediction:  4724.140625              actual:  4725
```

And we can try with other datasets

```
Office-Mac-mini:linearLearnerClient vincentbloise$ python client.py
prediction:  4153.671875              actual:  4153
```