Problem Statement and Domain

The problem presented is the is in the domain of health and medical practice where digitized images of a fine needle aspirate of a breast mass are examined for cancer. The diagnosis of the mass is malignant or benign. The diagnosis helps doctors find breast cancer in early stages and can improve the chances of survival of a patient if intervention is made early. This is necessary as it curbs the spread of the cancer cells to other parts of the body. In addition to the benefits of early detection, using machine learning and artificial intelligence for breast cancer detection is racially unbiased compared to traditional models

In [1]:

```
# Import all necessary modules
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import matplotlib
import sklearn.model_selection
%matplotlib inline
```

In [2]:

```
# sagemaker libraries
import boto3
import sagemaker
```

Dataset input

The dataset is obtained from https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data) which was originally created by Bennett, Kristin P., and Olvi L. Mangasarian. "Robust linear programming discrimination of two linearly inseparable sets." Optimization methods and software 1, no. 1 (1992): 23-34. The attribute information are as follows: 1) ID number 2) Diagnosis (M = malignant, B = benign) 3-32) Ten real-valued features are computed for each cell nucleus. Class distribution: 357 benign, 212 malignant

In [3]:

```
# Load the data and perform exploratory analysis
df = pd.read_csv('data.csv', header = 0, index_col = 0)
df.head()
```

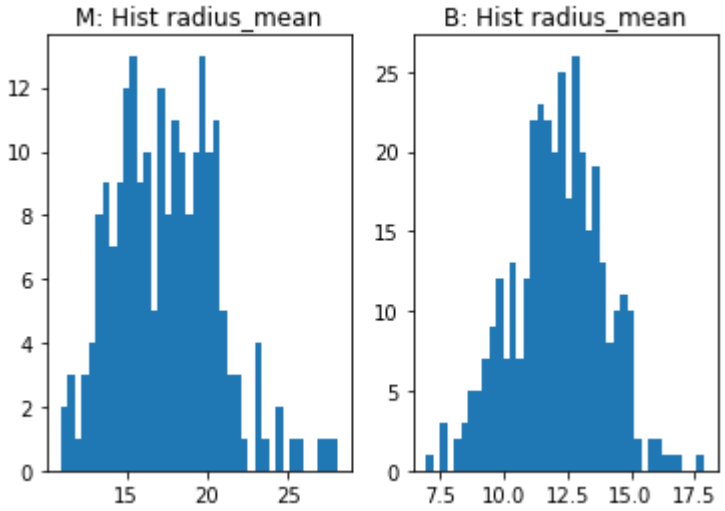
Out[3]:

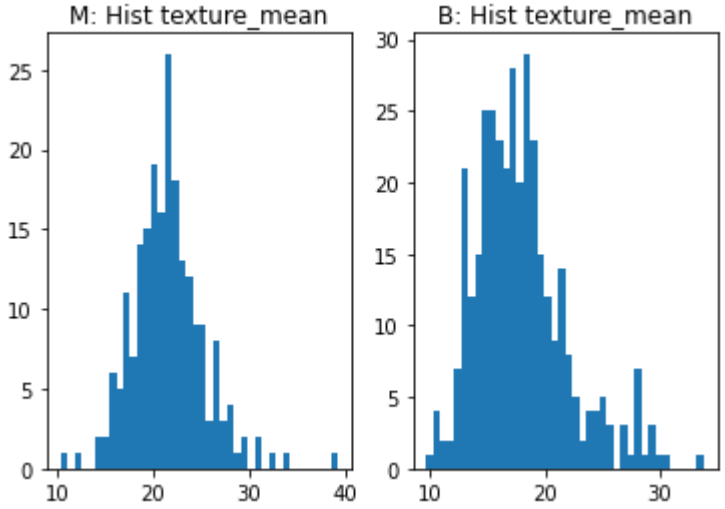
id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mear
842302	M	17.99	10.38	122.80	1001.0	0.1184(
842517	M	20.57	17.77	132.90	1326.0	0.08474
84300903	M	19.69	21.25	130.00	1203.0	0.1096(
84348301	M	11.42	20.38	77.58	386.1	0.1425(
84358402	M	20.29	14.34	135.10	1297.0	0.1003(

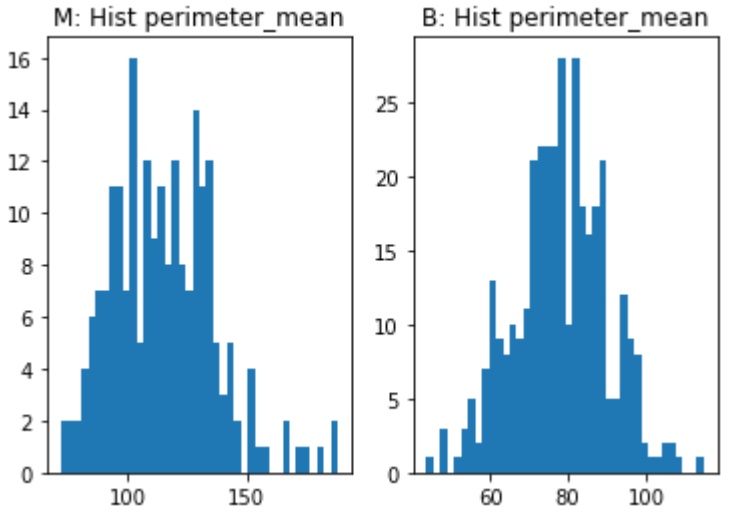
5 rows × 31 columns

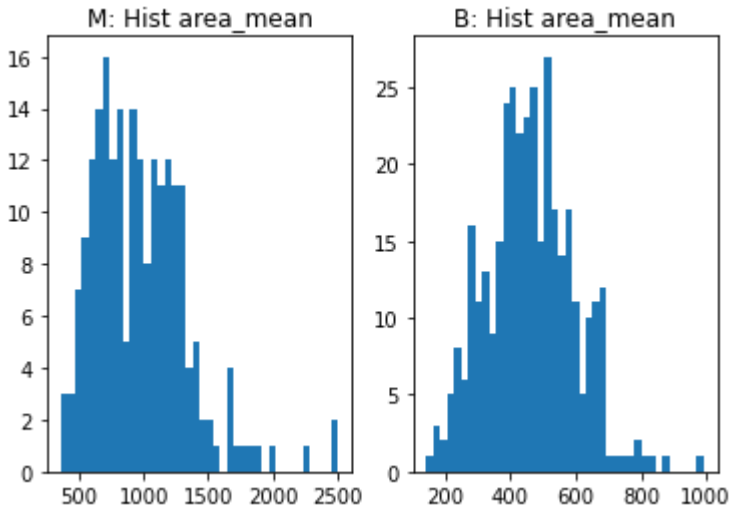
In [4]:

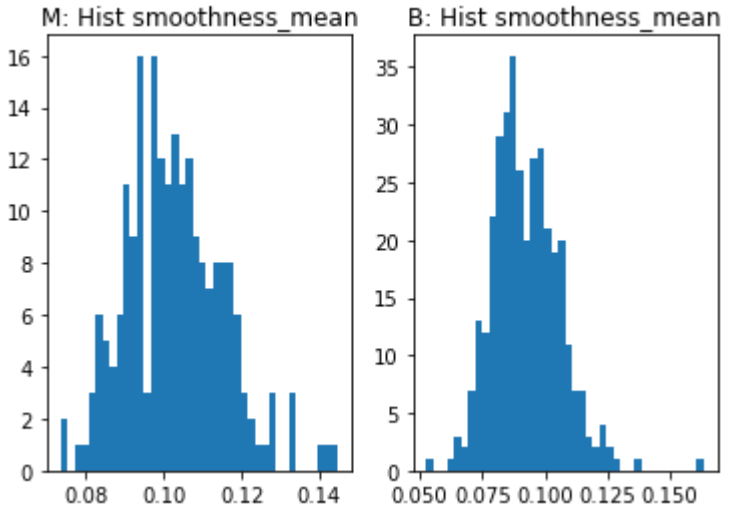
```
# Attributes
attribute_list = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothnes
s_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symmetry_mean', 'fr
actal_dimension_mean']
n_bins = 40

for column_name in attribute_list:
    fig,(ax1, ax2) =plt.subplots(1,2)
    # get data by column_name and display a histogram
    ax1.hist(df.loc[df['diagnosis'] == 'M', column_name], bins=n_bins)
    ax2.hist(df.loc[df['diagnosis'] == 'B', column_name], bins=n_bins)
    ax1.set_title( "M: Hist " + column_name)
    ax2.set_title( "B: Hist " + column_name)
```

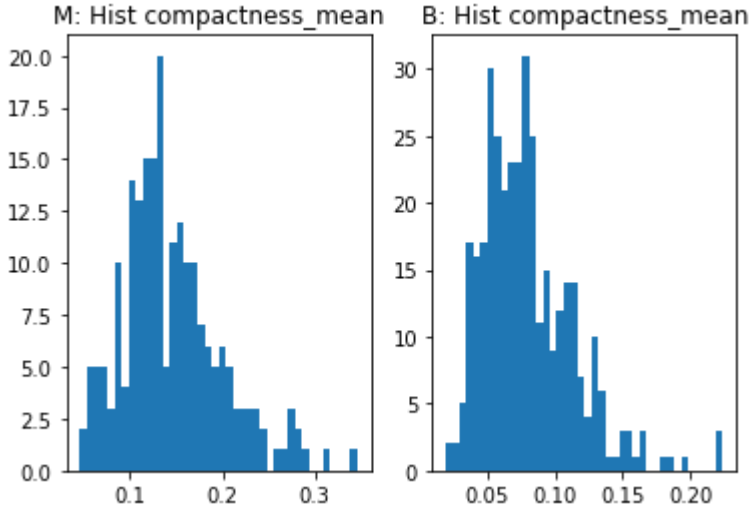

M: Hist radius_mean
B: Hist radius_mean

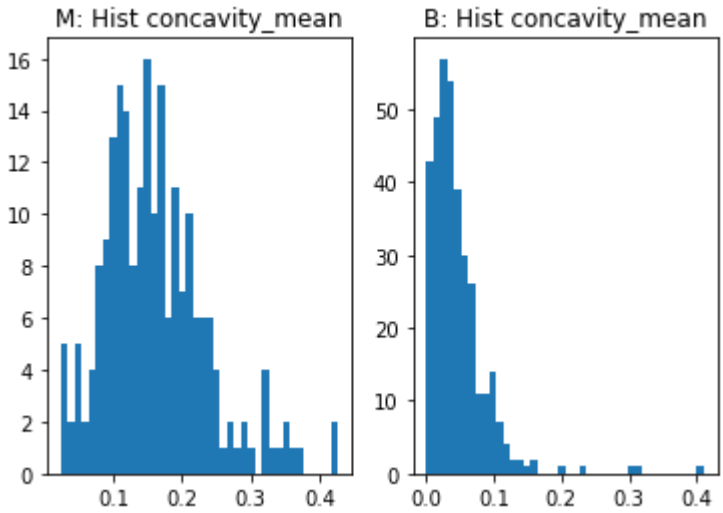

M: Hist texture_mean
B: Hist texture_mean

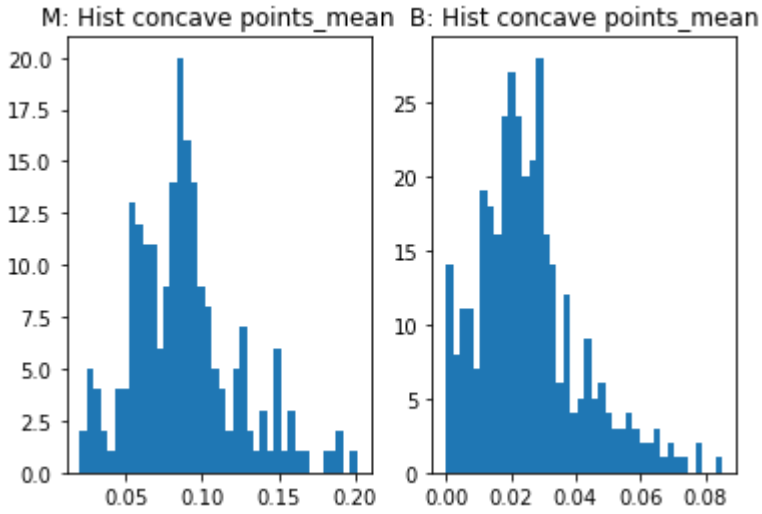

M: Hist perimeter_mean
B: Hist perimeter_mean

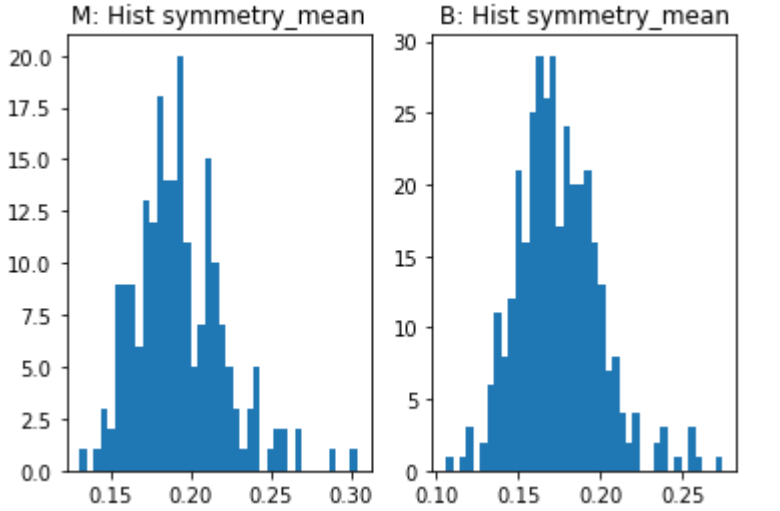

M: Hist area_mean
B: Hist area_mean

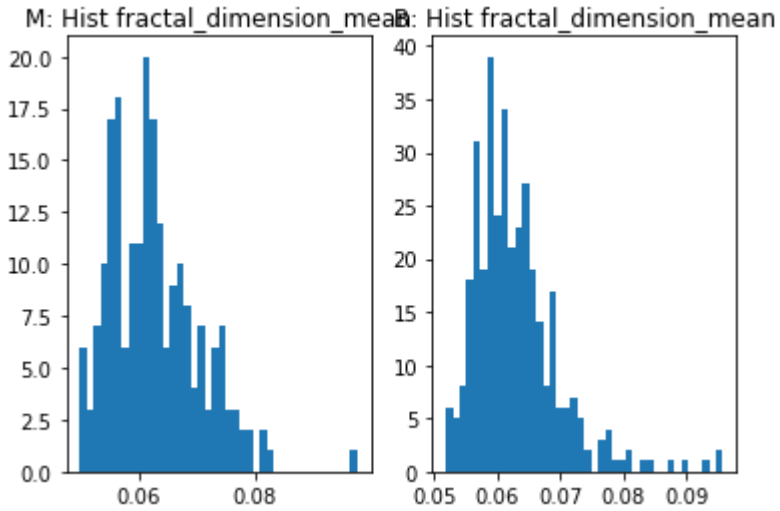

M: Hist smoothness_mean
B: Hist smoothness_mean


M: Hist compactness_mean
B: Hist compactness_mean


M: Hist concavity_mean
B: Hist concavity_mean


M: Hist concave points_mean
B: Hist concave points_mean





Exploratory Data Analysis The plots above show that a fair distinction between malignant and benign features. Chief of them being histconcave_points, radius and perimeter. A casual observation shows that a Linear classifier may do a pretty good job. However we will implement a Neural Network here in PyTorch and possibly a Support Vector Machine for comparison

In [5]:

```
from sagemaker import get_execution_role
session =sagemaker.Session()
# store the current SageMaker session
# get IAM role
role=get_execution_role()
print(role)

bucket_name=session.default_bucket()
print(bucket_name)

#define prefix and output path
prefix='mamogram'
output_path='s3://{}/{}/'.format(bucket_name,prefix)
```

```
arn:aws:iam::172268057478:role/service-role/AmazonSageMaker-ExecutionRole-202
10122T150167
sagemaker-us-east-1-172268057478
```

In [6]:

```
# Scale the data for normalization
from sklearn.preprocessing import MinMaxScaler
scaler  = MinMaxScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df.iloc[:,1:]), columns=df.iloc[:,1:].column
s)
```

Dimensionality Reduction

Given that we have many features present in this dataset, it is beneficial to reduce the number of features we will pass to our classifier. To do this, the principal component analysis from amazon sagemaker would be implemented as follows

In [7]:

```
from sagemaker import PCA

session = sagemaker.Session()

N_COMPONENTS = df_scaled.shape[1] - 1
pca_SM = PCA(role = role,
             train_instance_count = 1,
             train_instance_type = 'ml.c4.xlarge',
             output_path = output_path,
             num_components = N_COMPONENTS,
             sagemaker_session = session )

print(f' nr of components is {N_COMPONENTS}')

train_data_np = df.iloc[:,1:].values.astype('float32') # convert to float32 before record
 set
formatted_train_data = pca_SM.record_set( train_data_np )
```

```
train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.


 nr of components is 29
```

In [8]:

```
#Train the PCA model
%time
# train the PCA mode on the formatted data
pca_SM.fit(formatted_train_data)
```

```
Defaulting to the only supported framework/algorithm version: 1. Ignoring fra
mework/algorithm version: 1.
Defaulting to the only supported framework/algorithm version: 1. Ignoring fra
mework/algorithm version: 1.
```

```
CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 8.34 µs
2021-03-17 00:19:17 Starting - Starting the training job...
2021-03-17 00:19:41 Starting - Launching requested ML instancesProfilerReport
-1615940357: InProgress
......
2021-03-17 00:20:41 Starting - Preparing the instances for trainin
g...............
2021-03-17 00:23:12 Downloading - Downloading input data
2021-03-17 00:23:12 Training - Downloading the training image...
2021-03-17 00:23:44 Uploading - Uploading generated training modelDocker entr
ypoint called with argument(s): train
Running default environment configuration script
[03/17/2021 00:23:36 INFO 139991191717696] Reading default configuration from
/opt/amazon/lib/python2.7/site-packages/algorithm/resources/default-conf.jso
n: {u'_num_gpus': u'auto', u'_log_level': u'info', u'subtract_mean': u'true',
u'force_dense': u'true', u'epochs': 1, u'algorithm_mode': u'regular', u'extra
_components': u'-1', u'_kvstore': u'dist_sync', u'_num_kv_servers': u'auto'}
[03/17/2021 00:23:36 INFO 139991191717696] Merging with provided configuratio
n from /opt/ml/input/config/hyperparameters.json: {u'feature_dim': u'30', u'm
ini_batch_size': u'500', u'num_components': u'29'}
[03/17/2021 00:23:36 INFO 139991191717696] Final configuration: {u'num_compon
ents': u'29', u'_num_gpus': u'auto', u'_log_level': u'info', u'subtract_mea
n': u'true', u'force_dense': u'true', u'epochs': 1, u'algorithm_mode': u'regu
lar', u'feature_dim': u'30', u'extra_components': u'-1', u'_kvstore': u'dist_
sync', u'_num_kv_servers': u'auto', u'mini_batch_size': u'500'}
[03/17/2021 00:23:36 WARNING 139991191717696] Loggers have already been setu
p.
[03/17/2021 00:23:36 INFO 139991191717696] Launching parameter server for rol
e scheduler
[03/17/2021 00:23:36 INFO 139991191717696] {'ECS_CONTAINER_METADATA_URI': 'ht
tp://169.254.170.2/v3/1c107553-6bbc-41ea-a7f6-aba9c83c1222', 'ECS_CONTAINER_M
ETADATA_URI_V4': 'http://169.254.170.2/v4/1c107553-6bbc-41ea-a7f6-aba9c83c122
2', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION': '2', 'PATH': '/opt/amaz
on/bin:/usr/local/nvidia/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bi
n:/sbin:/bin:/opt/amazon/bin:/opt/amazon/bin', 'SAGEMAKER_HTTP_PORT': '8080',
'HOME': '/root', 'PYTHONUNBUFFERED': 'TRUE', 'CANONICAL_ENVROOT': '/opt/amazo
n', 'LD_LIBRARY_PATH': '/opt/amazon/lib/python2.7/site-packages/cv
2/../../../../lib:/usr/local/nvidia/lib64:/opt/amazon/lib', 'LANG': 'en_US.ut
f8', 'DMLC_INTERFACE': 'eth0', 'SHLVL': '1', 'AWS_REGION': 'us-east-1', 'NVID
IA_VISIBLE_DEVICES': 'void', 'TRAINING_JOB_NAME': 'pca-2021-03-17-00-19-17-23
2', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION': 'cpp', 'ENVROOT': '/opt/amazo
n', 'SAGEMAKER_DATA_PATH': '/opt/ml', 'NVIDIA_DRIVER_CAPABILITIES': 'compute,
utility', 'NVIDIA_REQUIRE_CUDA': 'cuda>=9.0', 'OMP_NUM_THREADS': '2', 'HOSTNA
ME': 'ip-10-2-156-221.ec2.internal', 'AWS_CONTAINER_CREDENTIALS_RELATIVE_UR
I': '/v2/credentials/f1e26f65-5a90-471c-9fa1-4c53596843d3', 'PWD': '/', 'TRAI
NING_JOB_ARN': 'arn:aws:sagemaker:us-east-1:172268057478:training-job/pca-202
1-03-17-00-19-17-232', 'AWS_EXECUTION_ENV': 'AWS_ECS_EC2'}
[03/17/2021 00:23:36 INFO 139991191717696] envs={'ECS_CONTAINER_METADATA_UR
I': 'http://169.254.170.2/v3/1c107553-6bbc-41ea-a7f6-aba9c83c1222', 'ECS_CONT
AINER_METADATA_URI_V4': 'http://169.254.170.2/v4/1c107553-6bbc-41ea-a7f6-aba9
c83c1222', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION': '2', 'DMLC_NUM_W
ORKER': '1', 'DMLC_PS_ROOT_PORT': '9000', 'PATH': '/opt/amazon/bin:/usr/loca
l/nvidia/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/op
t/amazon/bin:/opt/amazon/bin', 'SAGEMAKER_HTTP_PORT': '8080', 'HOME': '/roo
t', 'PYTHONUNBUFFERED': 'TRUE', 'CANONICAL_ENVROOT': '/opt/amazon', 'LD_LIBRA
RY_PATH': '/opt/amazon/lib/python2.7/site-packages/cv2/../../../../lib:/usr/l
```

ocal/nvidia/lib64:/opt/amazon/lib', 'LANG': 'en_US.utf8', 'DMLC_INTERFACE':
'eth0', 'SHLVL': '1', 'DMLC_PS_ROOT_URI': '10.2.156.221', 'AWS_REGION': 'us-
east-1', 'NVIDIA_VISIBLE_DEVICES': 'void', 'TRAINING_JOB_NAME': 'pca-2021-03-
17-00-19-17-232', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION': 'cpp', 'ENVROOT':
'/opt/amazon', 'SAGEMAKER_DATA_PATH': '/opt/ml', 'NVIDIA_DRIVER_CAPABILITIE
S': 'compute,utility', 'NVIDIA_REQUIRE_CUDA': 'cuda>=9.0', 'OMP_NUM_THREADS':
'2', 'HOSTNAME': 'ip-10-2-156-221.ec2.internal', 'AWS_CONTAINER_CREDENTIALS_R
ELATIVE_URI': '/v2/credentials/f1e26f65-5a90-471c-9fa1-4c53596843d3', 'DMLC_R
OLE': 'scheduler', 'PWD': '/', 'DMLC_NUM_SERVER': '1', 'TRAINING_JOB_ARN': 'a
rn:aws:sagemaker:us-east-1:172268057478:training-job/pca-2021-03-17-00-19-17-
232', 'AWS_EXECUTION_ENV': 'AWS_ECS_EC2'}
[03/17/2021 00:23:36 INFO 139991191717696] Launching parameter server for rol
e server
[03/17/2021 00:23:36 INFO 139991191717696] {'ECS_CONTAINER_METADATA_URI': 'ht
tp://169.254.170.2/v3/1c107553-6bbc-41ea-a7f6-aba9c83c1222', 'ECS_CONTAINER_M
ETADATA_URI_V4': 'http://169.254.170.2/v4/1c107553-6bbc-41ea-a7f6-aba9c83c122
2', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION': '2', 'PATH': '/opt/amaz
on/bin:/usr/local/nvidia/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bi
n:/sbin:/bin:/opt/amazon/bin:/opt/amazon/bin', 'SAGEMAKER_HTTP_PORT': '8080',
'HOME': '/root', 'PYTHONUNBUFFERED': 'TRUE', 'CANONICAL_ENVROOT': '/opt/amazo
n', 'LD_LIBRARY_PATH': '/opt/amazon/lib/python2.7/site-packages/cv
2/../../../../lib:/usr/local/nvidia/lib64:/opt/amazon/lib', 'LANG': 'en_US.ut
f8', 'DMLC_INTERFACE': 'eth0', 'SHLVL': '1', 'AWS_REGION': 'us-east-1', 'NVID
IA_VISIBLE_DEVICES': 'void', 'TRAINING_JOB_NAME': 'pca-2021-03-17-00-19-17-23
2', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION': 'cpp', 'ENVROOT': '/opt/amazo
n', 'SAGEMAKER_DATA_PATH': '/opt/ml', 'NVIDIA_DRIVER_CAPABILITIES': 'compute,
utility', 'NVIDIA_REQUIRE_CUDA': 'cuda>=9.0', 'OMP_NUM_THREADS': '2', 'HOSTNA
ME': 'ip-10-2-156-221.ec2.internal', 'AWS_CONTAINER_CREDENTIALS_RELATIVE_UR
I': '/v2/credentials/f1e26f65-5a90-471c-9fa1-4c53596843d3', 'PWD': '/', 'TRAI
NING_JOB_ARN': 'arn:aws:sagemaker:us-east-1:172268057478:training-job/pca-202
1-03-17-00-19-17-232', 'AWS_EXECUTION_ENV': 'AWS_ECS_EC2'}
[03/17/2021 00:23:36 INFO 139991191717696] envs={'ECS_CONTAINER_METADATA_UR
I': 'http://169.254.170.2/v3/1c107553-6bbc-41ea-a7f6-aba9c83c1222', 'ECS_CONT
AINER_METADATA_URI_V4': 'http://169.254.170.2/v4/1c107553-6bbc-41ea-a7f6-aba9
c83c1222', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION': '2', 'DMLC_NUM_W
ORKER': '1', 'DMLC_PS_ROOT_PORT': '9000', 'PATH': '/opt/amazon/bin:/usr/loca
l/nvidia/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/op
t/amazon/bin:/opt/amazon/bin', 'SAGEMAKER_HTTP_PORT': '8080', 'HOME': '/roo
t', 'PYTHONUNBUFFERED': 'TRUE', 'CANONICAL_ENVROOT': '/opt/amazon', 'LD_LIBRA
RY_PATH': '/opt/amazon/lib/python2.7/site-packages/cv2/../../../../lib:/usr/l
ocal/nvidia/lib64:/opt/amazon/lib', 'LANG': 'en_US.utf8', 'DMLC_INTERFACE':
'eth0', 'SHLVL': '1', 'DMLC_PS_ROOT_URI': '10.2.156.221', 'AWS_REGION': 'us-
east-1', 'NVIDIA_VISIBLE_DEVICES': 'void', 'TRAINING_JOB_NAME': 'pca-2021-03-
17-00-19-17-232', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION': 'cpp', 'ENVROOT':
'/opt/amazon', 'SAGEMAKER_DATA_PATH': '/opt/ml', 'NVIDIA_DRIVER_CAPABILITIE
S': 'compute,utility', 'NVIDIA_REQUIRE_CUDA': 'cuda>=9.0', 'OMP_NUM_THREADS':
'2', 'HOSTNAME': 'ip-10-2-156-221.ec2.internal', 'AWS_CONTAINER_CREDENTIALS_R
ELATIVE_URI': '/v2/credentials/f1e26f65-5a90-471c-9fa1-4c53596843d3', 'DMLC_R
OLE': 'server', 'PWD': '/', 'DMLC_NUM_SERVER': '1', 'TRAINING_JOB_ARN': 'arn:
aws:sagemaker:us-east-1:172268057478:training-job/pca-2021-03-17-00-19-17-23
2', 'AWS_EXECUTION_ENV': 'AWS_ECS_EC2'}
[03/17/2021 00:23:36 INFO 139991191717696] Environment: {'ECS_CONTAINER_METAD
ATA_URI': 'http://169.254.170.2/v3/1c107553-6bbc-41ea-a7f6-aba9c83c1222', 'EC
S_CONTAINER_METADATA_URI_V4': 'http://169.254.170.2/v4/1c107553-6bbc-41ea-a7f
6-aba9c83c1222', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION': '2', 'DMLC
_PS_ROOT_PORT': '9000', 'DMLC_NUM_WORKER': '1', 'SAGEMAKER_HTTP_PORT': '808
0', 'PATH': '/opt/amazon/bin:/usr/local/nvidia/bin:/usr/local/sbin:/usr/loca

```
l/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/amazon/bin:/opt/amazon/bin', 'PYTHON
UNBUFFERED': 'TRUE', 'CANONICAL_ENVROOT': '/opt/amazon', 'LD_LIBRARY_PATH':
 '/opt/amazon/lib/python2.7/site-packages/cv2/../../../../lib:/usr/local/nvid
ia/lib64:/opt/amazon/lib', 'LANG': 'en_US.utf8', 'DMLC_INTERFACE': 'eth0', 'S
HLVL': '1', 'DMLC_PS_ROOT_URI': '10.2.156.221', 'AWS_REGION': 'us-east-1', 'N
VIDIA_VISIBLE_DEVICES': 'void', 'TRAINING_JOB_NAME': 'pca-2021-03-17-00-19-17
-232', 'HOME': '/root', 'PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION': 'cpp', 'ENV
ROOT': '/opt/amazon', 'SAGEMAKER_DATA_PATH': '/opt/ml', 'NVIDIA_DRIVER_CAPABI
LITIES': 'compute,utility', 'NVIDIA_REQUIRE_CUDA': 'cuda>=9.0', 'OMP_NUM_THRE
ADS': '2', 'HOSTNAME': 'ip-10-2-156-221.ec2.internal', 'AWS_CONTAINER_CREDENT
IALS_RELATIVE_URI': '/v2/credentials/f1e26f65-5a90-471c-9fa1-4c53596843d3',
 'DMLC_ROLE': 'worker', 'PWD': '/', 'DMLC_NUM_SERVER': '1', 'TRAINING_JOB_AR
N': 'arn:aws:sagemaker:us-east-1:172268057478:training-job/pca-2021-03-17-00-
19-17-232', 'AWS_EXECUTION_ENV': 'AWS_ECS_EC2'}
Process 61 is a shell:scheduler.
Process 70 is a shell:server.
Process 1 is a worker.
[03/17/2021 00:23:36 INFO 139991191717696] Using default worker.
[03/17/2021 00:23:36 INFO 139991191717696] Loaded iterator creator applicatio
n/x-recordio-protobuf for content type ('application/x-recordio-protobuf',
 '1.0')
[03/17/2021 00:23:36 INFO 139991191717696] Loaded iterator creator applicatio
n/x-labeled-vector-protobuf for content type ('application/x-labeled-vector-p
rotobuf', '1.0')
[03/17/2021 00:23:36 INFO 139991191717696] Loaded iterator creator protobuf f
or content type ('protobuf', '1.0')
[03/17/2021 00:23:36 INFO 139991191717696] Checkpoint loading and saving are
 disabled.
[03/17/2021 00:23:36 INFO 139991191717696] Create Store: dist_sync
[03/17/2021 00:23:37 INFO 139991191717696] nvidia-smi took: 0.025209903717 se
cs to identify 0 gpus
[03/17/2021 00:23:37 INFO 139991191717696] Number of GPUs being used: 0
[03/17/2021 00:23:37 INFO 139991191717696] The default executor is <PCAExecut
or on cpu(0)>.
[03/17/2021 00:23:37 INFO 139991191717696] 30 feature(s) found in 'data'.
[03/17/2021 00:23:37 INFO 139991191717696] <PCAExecutor on cpu(0)> is assigne
d to batch slice from 0 to 499.
#metrics {"Metrics": {"initialize.time": {"count": 1, "max": 560.767173767089
8, "sum": 560.7671737670898, "min": 560.7671737670898}}, "EndTime": 161594061
7.391637, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorith
m": "PCA"}, "StartTime": 1615940616.824652}

#metrics {"Metrics": {"Max Batches Seen Between Resets": {"count": 1, "max":
 0, "sum": 0.0, "min": 0}, "Number of Batches Since Last Reset": {"count": 1,
"max": 0, "sum": 0.0, "min": 0}, "Number of Records Since Last Reset": {"coun
t": 1, "max": 0, "sum": 0.0, "min": 0}, "Total Batches Seen": {"count": 1, "m
ax": 0, "sum": 0.0, "min": 0}, "Total Records Seen": {"count": 1, "max": 0,
 "sum": 0.0, "min": 0}, "Max Records Seen Between Resets": {"count": 1, "ma
x": 0, "sum": 0.0, "min": 0}, "Reset Count": {"count": 1, "max": 0, "sum": 0.
0, "min": 0}}, "EndTime": 1615940617.391885, "Dimensions": {"Host": "algo-1",
"Meta": "init_train_data_iter", "Operation": "training", "Algorithm": "PCA"},
"StartTime": 1615940617.391839}

[2021-03-17 00:23:37.392] [tensorio] [info] epoch_stats={"data_pipeline": "/o
pt/ml/input/data/train", "epoch": 0, "duration": 564, "num_examples": 1, "num
_bytes": 74000}
[2021-03-17 00:23:37.456] [tensorio] [info] epoch_stats={"data_pipeline": "/o
```

pt/ml/input/data/train", "epoch": 1, "duration": 54, "num_examples": 2, "num_
bytes": 84212}
#metrics {"Metrics": {"epochs": {"count": 1, "max": 1, "sum": 1.0, "min": 1},
"update.time": {"count": 1, "max": 64.7430419921875, "sum": 64.7430419921875,
"min": 64.7430419921875}}, "EndTime": 1615940617.457065, "Dimensions": {"Hos
t": "algo-1", "Operation": "training", "Algorithm": "PCA"}, "StartTime": 1615
940617.391745}

[03/17/2021 00:23:37 INFO 139991191717696] #progress_metric: host=algo-1, com
pleted 100 % of epochs
#metrics {"Metrics": {"Max Batches Seen Between Resets": {"count": 1, "max":
 2, "sum": 2.0, "min": 2}, "Number of Batches Since Last Reset": {"count": 1,
"max": 2, "sum": 2.0, "min": 2}, "Number of Records Since Last Reset": {"coun
t": 1, "max": 569, "sum": 569.0, "min": 569}, "Total Batches Seen": {"count":
1, "max": 2, "sum": 2.0, "min": 2}, "Total Records Seen": {"count": 1, "max":
569, "sum": 569.0, "min": 569}, "Max Records Seen Between Resets": {"count":
 1, "max": 569, "sum": 569.0, "min": 569}, "Reset Count": {"count": 1, "max":
1, "sum": 1.0, "min": 1}}, "EndTime": 1615940617.457711, "Dimensions": {"Hos
t": "algo-1", "Meta": "training_data_iter", "Operation": "training", "Algorit
hm": "PCA", "epoch": 0}, "StartTime": 1615940617.392272}

[03/17/2021 00:23:37 INFO 139991191717696] #throughput_metric: host=algo-1, t
rain throughput=8674.04348363 records/second
#metrics {"Metrics": {"finalize.time": {"count": 1, "max": 39.89815711975098,
"sum": 39.89815711975098, "min": 39.89815711975098}}, "EndTime": 1615940617.4
98077, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm":
"PCA"}, "StartTime": 1615940617.457347}

[03/17/2021 00:23:37 INFO 139991191717696] Test data is not provided.
#metrics {"Metrics": {"totaltime": {"count": 1, "max": 866.779088973999, "su
m": 866.779088973999, "min": 866.779088973999}, "setuptime": {"count": 1, "ma
x": 20.46799659729004, "sum": 20.46799659729004, "min": 20.46799659729004}},
 "EndTime": 1615940617.513222, "Dimensions": {"Host": "algo-1", "Operation":
 "training", "Algorithm": "PCA"}, "StartTime": 1615940617.498194}

2021-03-17 00:24:04 Completed - Training job completed
Training seconds: 51
Billable seconds: 51

In [9]:

```
training_job_name='pca-2021-03-17-00-19-17-232'
model_key=os.path.join(prefix,training_job_name,'output/model.tar.gz')

print(model_key)
# download and unzip model
boto3.resource('s3').Bucket(bucket_name).download_file(model_key,'model.tar.gz')
# unzipping as model_algo-1
os.system('tar -zxvf model.tar.gz')
os.system('unzip model_algo-1')
```

mamogram/pca-2021-03-17-00-19-17-232/output/model.tar.gz

Out[9]:

2304

Select Nr of components to retain using explained variance

We will select a number of components that retain at least 85% of the explained variance in the original data

In [10]:

```
import mxnet as mx
# loading the unzipped artifacts
pca_model_params=mx.ndarray.load('model_algo-1')

# get selected params
s=pd.DataFrame(pca_model_params['s'].asnumpy())
```

In [11]:

```
# source : Population Segmentation exercise notebokk Udacity
def explained_variance(s , n_top_components ):
    '''Calculates the approx. data variance that n_top_components captures.
    :param s: A dataframe of singular values for top components;
    the top value is in the last row.
    :param n_top_components: An integer, the number of top components to use.
    :return: The expected data variance covered by the n_top_components.'''
    start_idx = N_COMPONENTS - n_top_components
    # calculate approx variance
    exp_variance = np.square(s.iloc[start_idx:,:]).sum()/np.square(s).sum()
    return exp_variance[0]
```

In [12]:

```
# test cell
#print(s)
n_top_components = N_COMPONENTS//2
# select a set of values for the number of top components and examing the corresponding ex
plained variance
# calculate the explained variance


print('----------------------------------------\n')
print('n_top_component   ', 'Explained variance: ')
for top_component in range(1,n_top_components):
    exp_variance = explained_variance ( s , top_component )
    print ( ' %-14d     %-9.6f %%' %(top_component, exp_variance*100 ) )
```

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

n_top_component    Explained variance:
 1                  98.204499 %
 2                  99.822116 %
 3                  99.977863 %
 4                  99.989957 %
 5                  99.998790 %
 6                  99.999452 %
 7                  99.999857 %
 8                  99.999928 %
 9                  99.999970 %
 10                 99.999988 %
 11                 99.999994 %
 12                 100.000000 %
 13                 100.000000 %
```

Nr of components to choose : The analysis above shows that we can get away with 5-6 features and yet retain a huge percentage of the variance in the orginial dataset

In [13]:

```
# Deploy the pca
%time
pca_predictor = pca_SM.deploy(initial_instance_count=1,instance_type='ml.t2.medium')
```

```
Defaulting to the only supported framework/algorithm version: 1. Ignoring fra
mework/algorithm version: 1.

CPU times: user 4 µs, sys: 0 ns, total: 4 µs
Wall time: 9.78 µs
-------------------!
```

In [14]:

```
# pass np train data to the PCA model
train_pca = pca_predictor.predict(train_data_np)
```

In [15]:

```
# source: Population Segmentation exercise notebokk Udacity
# create dimensionality-reduced data
def create_transformed_df( train_pca , df_scaled , n_top_components ):
    ''' Return a dataframe of data points with component features.
    The dataframe should be indexed by id and contain component values.
    :param train_pca: A list of pca training data, returned by a PCA model.
    :param df_scaled: A dataframe of normalized, original features.
    :param n_top_components: An integer, the number of top components to use.
    :return: A dataframe, indexed by id, with n_top_component values as columns.
    '''
    # create new dataframe to add data to
    df_transformed = pd.DataFrame()
    # for each of our new, transformed data points # append the component values to the da
taframe
    for data in train_pca:
        # get component values for each data point
        components = data.label['projection'].float32_tensor.values
        df_transformed = df_transformed.append([list(components)])

    df_transformed.index=df_scaled.index # index by id as in the previous dataframe

    # keep only the top n components
    start_idx = N_COMPONENTS-n_top_components
    df_transformed=df_transformed.iloc[:,start_idx:]

    # reverse columns, component order
    return df_transformed.iloc[:, ::-1]
```

In [24]:

```
# specify top n
top_n = 5
# call your function and create a new dataframe
df_transformed = create_transformed_df(train_pca, df_scaled, n_top_components = top_n)

# rename columns
PCA_list = ['c_' + str(i) for i in range(1,top_n+1)]

df_transformed.columns = PCA_list

# print result
df_transformed.head()
```

Out[24]:

	c_1	c_2	c_3	c_4	c_5
0	1160.142456	-293.917755	48.578362	8.711304	-32.000587
1	1269.122437	15.630157	-35.394497	-17.861328	4.335159
2	995.793823	39.156708	-1.709747	-4.199310	0.466625
3	-407.180725	-67.380241	8.672813	11.759758	-7.115633
4	930.341309	189.340622	1.374796	-8.499107	-7.613194

Prepare the data for training

We will use 6 features from the reduced dimensionality produced by the PCA Split the data into test train : 67% : 33% Store the training and test data in csv and upload to s3

In [29]:

```
dict_category = {'B':0, 'M':1}
target = df.iloc[:,0].apply(lambda x : dict_category[x]) # convert the class into 0-1 Bina
ry and store the target class
X_train,X_test,Y_train,Y_test = sklearn.model_selection.train_test_split(df_transformed, t
arget,test_size=0.2)
```

In [26]:

```
df_transformed.head()
```

Out[26]:

	c_1	c_2	c_3	c_4	c_5
0	1160.142456	-293.917755	48.578362	8.711304	-32.000587
1	1269.122437	15.630157	-35.394497	-17.861328	4.335159
2	995.793823	39.156708	-1.709747	-4.199310	0.466625
3	-407.180725	-67.380241	8.672813	11.759758	-7.115633
4	930.341309	189.340622	1.374796	-8.499107	-7.613194

In [19]:

```
target.head()
```

Out[19]:

```
id
842302      1
842517      1
84300903    1
84348301    1
84358402    1
Name: diagnosis, dtype: int64
```

In [27]:

```
# This is our local data directory. We need to make sure that it exists.
data_dir = '../Capstone Project/data/'
if not os.path.exists( data_dir ):
    os.makedirs(data_dir)
```

In [30]:

```
# create the training and test data and save locally
test = np.hstack([np.reshape(Y_test.values, (Y_test.shape[0], 1)), X_test.values])
train = np.hstack([np.reshape(Y_train.values, (Y_train.shape[0], 1)), X_train.values])
pd.DataFrame(test).to_csv(os.path.join(data_dir,'test.csv'),header=False,index=False)
pd.DataFrame(train).to_csv(os.path.join(data_dir,'train.csv'),header=False,index=False)
```

In [22]:

```
# set prefix, a descriptive name for a directory for our train test data
prefix = 'cancer-class'
# upload all data to S3
test_location = session.upload_data(os.path.join(data_dir,'test.csv'),key_prefix=prefix)
train_location= session.upload_data(os.path.join(data_dir,'train.csv'),key_prefix=prefix)
```

In [23]:

```
# Create correlation matrix for just Features to determine different models to test
corr_matrix = df_transformed.corr().abs().round(2)

# display shows all of a dataframe
display(corr_matrix)
```

	c_1	c_2	c_3	c_4	c_5	c_6
c_1	1.0	0.0	0.0	0.0	0.0	0.0
c_2	0.0	1.0	0.0	0.0	0.0	0.0
c_3	0.0	0.0	1.0	0.0	0.0	0.0
c_4	0.0	0.0	0.0	1.0	0.0	0.0
c_5	0.0	0.0	0.0	0.0	1.0	0.0
c_6	0.0	0.0	0.0	0.0	0.0	1.0

In [ ]:

```

```