In [82]:

```
import pandas as pd
import sklearn as sk
import numpy as np
import sagemaker as sg
import matplotlib as mt
from sklearn.model_selection import train_test_split
```

In [83]:

```
import os
import boto3
import re
import sagemaker

role = sagemaker.get_execution_role()
region = boto3.Session().region_name

# S3 bucket for saving code and model artifacts.
# Feel free to specify a different bucket and prefix
bucket = sagemaker.Session().default_bucket()

prefix = (
    "sagemaker/Diabetes-Pred"  # place to upload training files within the bucket
)
```

In [84]:

```
 s3 = boto3.client("s3")
data = pd.read_csv('https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv', header = None)
data.to_csv("data.csv", sep=",", index=False)
```

In [85]:

```
data.columns = ["number_times_pregnant", "plasma_glucose", "diastolic_blood_pressure", "triceps_skin_fold", "serum_insulin", "body_mass_index", "pedigree_function", "age", "will_get_diabetes"]
```

In [5]:

```
data.head()
```

Out[5]:

| | number_times_pregnant | plasma_glucose | diastolic_blood_pressure | triceps_skin_fold | serum_in |
|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | |
| **1** | 1 | 85 | 66 | 29 | |
| **2** | 8 | 183 | 64 | 0 | |
| **3** | 1 | 89 | 66 | 23 | |
| **4** | 0 | 137 | 40 | 35 | |

In [8]:

```
data.describe()
```

Out[8]:

| | number_times_pregnant | plasma_glucose | diastolic_blood_pressure | triceps_skin_fold | seru |
|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 7 |
| **mean** | 3.845052 | 120.894531 | 69.105469 | 20.536458 | |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 1 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 1 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 8 |

In [9]:

```
x= data.drop(columns="will_get_diabetes",axis=1, inplace= False)
```

In [10]:

```
y=data["will_get_diabetes"]
```

In [11]:

```
x_train,x_valid,y_train,y_valid= train_test_split(x,y,test_size=0.20)
```

In [12]:

```
#import module in terms of dealing with various types of I/O
import io
#import sagemaker common library
import sagemaker.amazon.common as smac
#converts the data in numpy array format to RecordIO format
buf = io.BytesIO()
smac.write_numpy_to_dense_tensor(buf, np.array(x_train).astype('float32'), np.array(y_trai
n).astype('float32'))

#reset in-memory byte arrays to zero
buf.seek(0)

key = 'linearlearner'
boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train', key)).upload_file
obj(buf)
s3_train_data = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
print('uploaded training data location: {}'.format(s3_train_data))

output_location = 's3://{}/{}/output'.format(bucket, prefix)
print('training artifacts will be uploaded to: {}'.format(output_location))
```

```
uploaded training data location: s3://sagemaker-us-east-1-412013648313/sagema
ker/Diabetes-Pred/train/linearlearner
training artifacts will be uploaded to: s3://sagemaker-us-east-1-41201364831
3/sagemaker/Diabetes-Pred/output
```

In [13]:

```
###Uploading test data
buf = io.BytesIO() # create an in-memory byte array (buf is a buffer I will be writing to)
smac.write_numpy_to_dense_tensor(buf, np.array(x_valid).astype('float32'), np.array(y_vali
d).astype('float32'))
buf.seek(0)

#Sub-folder for test data
key = 'linear-test-data'
boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'valid', key)).upload_file
obj(buf)
s3_test_data = 's3://{}/{}/valid/{}'.format(bucket, prefix, key)
print('uploaded validating data location: {}'.format(s3_test_data))

###Model Artifacts
output_location = 's3://{}/{}/output'.format(bucket, prefix)
print('Training artifacts will be uploaded to: {}'.format(output_location))
```

```
uploaded validating data location: s3://sagemaker-us-east-1-412013648313/sage
maker/Diabetes-Pred/valid/linear-test-data
Training artifacts will be uploaded to: s3://sagemaker-us-east-1-41201364831
3/sagemaker/Diabetes-Pred/output
```

In [14]:

```
 containers = {
              'us-east-1': '382416733822.dkr.ecr.us-east-1.amazonaws.com/linear-learner:la
test'
              }
```

In [15]:

```
sess = sagemaker.Session()
linear = sagemaker.estimator.Estimator(containers[boto3.Session().region_name],
                                       role=role,
                                       instance_count=1,
                                       instance_type='ml.m5.large',
                                       output_path=output_location,
                                       sagemaker_session=sess)
```

In [16]:

```
%%time


linear.set_hyperparameters(feature_dim=8,
                           mini_batch_size=100,
                           predictor_type='binary_classifier')
linear.fit({'train': s3_train_data})
```

```
2021-10-12 13:41:04 Starting - Starting the training job...
2021-10-12 13:41:28 Starting - Launching requested ML instancesProfilerReport
-1634046064: InProgress
......
2021-10-12 13:42:28 Starting - Preparing the instances for training.........
2021-10-12 13:43:48 Downloading - Downloading input data...
2021-10-12 13:44:29 Training - Downloading the training image.Docker entrypoi
nt called with argument(s): train
Running default environment configuration script
[10/12/2021 13:44:31 INFO 140550432450368] Reading default configuration from
/opt/amazon/lib/python3.7/site-packages/algorithm/resources/default-input.jso
n: {'mini_batch_size': '1000', 'epochs': '15', 'feature_dim': 'auto', 'use_bi
as': 'true', 'binary_classifier_model_selection_criteria': 'accuracy', 'f_bet
a': '1.0', 'target_recall': '0.8', 'target_precision': '0.8', 'num_models':
 'auto', 'num_calibration_samples': '10000000', 'init_method': 'uniform', 'in
it_scale': '0.07', 'init_sigma': '0.01', 'init_bias': '0.0', 'optimizer': 'au
to', 'loss': 'auto', 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivit
y': '0.01', 'huber_delta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3',
 'wd': 'auto', 'l1': 'auto', 'momentum': 'auto', 'learning_rate': 'auto', 'be
ta_1': 'auto', 'beta_2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mult': 'aut
o', 'use_lr_scheduler': 'true', 'lr_scheduler_step': 'auto', 'lr_scheduler_fa
ctor': 'auto', 'lr_scheduler_minimum_lr': 'auto', 'positive_example_weight_mu
lt': '1.0', 'balance_multiclass_weights': 'false', 'normalize_data': 'true',
 'normalize_label': 'auto', 'unbias_data': 'auto', 'unbias_label': 'auto', 'n
um_point_for_scaler': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num
_kv_servers': 'auto', '_log_level': 'info', '_tuning_objective_metric': '',
 'early_stopping_patience': '3', 'early_stopping_tolerance': '0.001', '_enabl
e_profiler': 'false'}
[10/12/2021 13:44:31 INFO 140550432450368] Merging with provided configuratio
n from /opt/ml/input/config/hyperparameters.json: {'feature_dim': '8', 'predi
ctor_type': 'binary_classifier', 'mini_batch_size': '100'}
[10/12/2021 13:44:31 INFO 140550432450368] Final configuration: {'mini_batch_
size': '100', 'epochs': '15', 'feature_dim': '8', 'use_bias': 'true', 'binary
_classifier_model_selection_criteria': 'accuracy', 'f_beta': '1.0', 'target_r
ecall': '0.8', 'target_precision': '0.8', 'num_models': 'auto', 'num_calibrat
ion_samples': '10000000', 'init_method': 'uniform', 'init_scale': '0.07', 'in
it_sigma': '0.01', 'init_bias': '0.0', 'optimizer': 'auto', 'loss': 'auto',
 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivity': '0.01', 'huber_del
ta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3', 'wd': 'auto', 'l1': 'a
uto', 'momentum': 'auto', 'learning_rate': 'auto', 'beta_1': 'auto', 'beta_
2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mult': 'auto', 'use_lr_schedule
r': 'true', 'lr_scheduler_step': 'auto', 'lr_scheduler_factor': 'auto', 'lr_s
cheduler_minimum_lr': 'auto', 'positive_example_weight_mult': '1.0', 'balance
_multiclass_weights': 'false', 'normalize_data': 'true', 'normalize_label':
 'auto', 'unbias_data': 'auto', 'unbias_label': 'auto', 'num_point_for_scale
r': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num_kv_servers': 'aut
o', '_log_level': 'info', '_tuning_objective_metric': '', 'early_stopping_pat
ience': '3', 'early_stopping_tolerance': '0.001', '_enable_profiler': 'fals
e', 'predictor_type': 'binary_classifier'}
[10/12/2021 13:44:31 WARNING 140550432450368] Loggers have already been setu
p.
Process 1 is a worker.
[10/12/2021 13:44:31 INFO 140550432450368] Using default worker.
[10/12/2021 13:44:31 INFO 140550432450368] Checkpoint loading and saving are
 disabled.
[2021-10-12 13:44:31.755] [tensorio] [info] epoch_stats={"data_pipeline": "/o
pt/ml/input/data/train", "epoch": 0, "duration": 13, "num_examples": 1, "num_
```

#metrics {"StartTime": 1634046274.7787194, "EndTime": 1634046274.7787297, "Dimensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "training", "epoch": 12, "model": 29}, "Metrics": {"train_binary_classification_cross_entropy_objective": {"sum": 0.6527500915527343, "count": 1, "min": 0.6527500915527343, "max": 0.6527500915527343}}}

#metrics {"StartTime": 1634046274.7787635, "EndTime": 1634046274.7787716, "Dimensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "training", "epoch": 12, "model": 30}, "Metrics": {"train_binary_classification_cross_entropy_objective": {"sum": 0.6521945889790853, "count": 1, "min": 0.6521945889790853, "max": 0.6521945889790853}}}

#metrics {"StartTime": 1634046274.778801, "EndTime": 1634046274.7788086, "Dimensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "training", "epoch": 12, "model": 31}, "Metrics": {"train_binary_classification_cross_entropy_objective": {"sum": 0.6524750328063965, "count": 1, "min": 0.6524750328063965, "max": 0.6524750328063965}}}

[10/12/2021 13:44:34 INFO 140550432450368] #quality_metric: host=algo-1, epoch=12, train binary_classification_cross_entropy_objective <loss>=0.5167867279052735
[10/12/2021 13:44:34 INFO 140550432450368] #early_stopping_criteria_metric: host=algo-1, epoch=12, criteria=binary_classification_cross_entropy_objective, value=0.47254947662353514
[10/12/2021 13:44:34 INFO 140550432450368] Saving model for epoch: 12
[10/12/2021 13:44:34 INFO 140550432450368] Saved checkpoint to "/tmp/tmp81g_vgme/mx-mod-0000.params"
[10/12/2021 13:44:34 INFO 140550432450368] Early stop condition met. Stopping training.
[10/12/2021 13:44:34 INFO 140550432450368] #progress_metric: host=algo-1, completed 100 % epochs
#metrics {"StartTime": 1634046274.5503454, "EndTime": 1634046274.788062, "Dimensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "training", "epoch": 12, "Meta": "training_data_iter"}, "Metrics": {"Total Records Seen": {"sum": 8696.0, "count": 1, "min": 8696, "max": 8696}, "Total Batches Seen": {"sum": 99.0, "count": 1, "min": 99, "max": 99}, "Max Records Seen Between Resets": {"sum": 614.0, "count": 1, "min": 614, "max": 614}, "Max Batches Seen Between Resets": {"sum": 7.0, "count": 1, "min": 7, "max": 7}, "Reset Count": {"sum": 15.0, "count": 1, "min": 15, "max": 15}, "Number of Records Since Last Reset": {"sum": 614.0, "count": 1, "min": 614, "max": 614}, "Number of Batches Since Last Reset": {"sum": 7.0, "count": 1, "min": 7, "max": 7}}}

[10/12/2021 13:44:34 INFO 140550432450368] #throughput_metric: host=algo-1, train throughput=2581.2757280346723 records/second
[10/12/2021 13:44:34 WARNING 140550432450368] wait_for_all_workers will not sync workers since the kv store is not running distributed
[10/12/2021 13:44:34 WARNING 140550432450368] wait_for_all_workers will not sync workers since the kv store is not running distributed
[2021-10-12 13:44:34.789] [tensorio] [info] epoch_stats={"data_pipeline": "/opt/ml/input/data/train", "epoch": 30, "duration": 0, "num_examples": 1, "num_bytes": 7600}
[2021-10-12 13:44:35.015] [tensorio] [info] epoch_stats={"data_pipeline": "/opt/ml/input/data/train", "epoch": 33, "duration": 223, "num_examples": 7, "num_bytes": 46664}
[2021-10-12 13:44:35.058] [tensorio] [info] epoch_stats={"data_pipeline": "/o

pt/ml/input/data/train", "epoch": 35, "duration": 22, "num_examples": 7, "num
_bytes": 46664}
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('binary_c
lassification_cross_entropy_objective', 0.46392177215228253)
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('binary_c
lassification_accuracy', 0.7915309446254072)
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('binary_f
_1.000', 0.681592039800995)
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('precisio
n', 0.7210526315789474)
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('recall',
0.6462264150943396)
[10/12/2021 13:44:35 INFO 140550432450368] #train_score (algo-1) : ('roc_auc_
score', 0.8422862104571482)
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n binary_classification_cross_entropy_objective <loss>=0.46392177215228253
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n binary_classification_accuracy <score>=0.7915309446254072
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n binary_f_1.000 <score>=0.681592039800995
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n precision <score>=0.7210526315789474
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n recall <score>=0.6462264150943396
[10/12/2021 13:44:35 INFO 140550432450368] #quality_metric: host=algo-1, trai
n roc_auc_score <score>=0.8422862104571482
[10/12/2021 13:44:35 INFO 140550432450368] Best model found for hyperparamete
rs: {"optimizer": "adam", "learning_rate": 0.1, "wd": 0.0001, "l1": 0.0, "lr_
scheduler_step": 10, "lr_scheduler_factor": 0.99, "lr_scheduler_minimum_lr":
 1e-05}
[10/12/2021 13:44:35 INFO 140550432450368] Saved checkpoint to "/tmp/tmp09uhz
yux/mx-mod-0000.params"
[10/12/2021 13:44:35 INFO 140550432450368] Test data is not provided.
#metrics {"StartTime": 1634046271.7411218, "EndTime": 1634046275.0702195, "Di
mensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "tr
aining"}, "Metrics": {"initialize.time": {"sum": 150.41589736938477, "count":
1, "min": 150.41589736938477, "max": 150.41589736938477}, "epochs": {"sum": 1
5.0, "count": 1, "min": 15, "max": 15}, "check_early_stopping.time": {"sum":
 8.453130722045898, "count": 13, "min": 0.2295970916748047, "max": 1.35755538
94042969}, "update.time": {"sum": 2844.05517578125, "count": 13, "min": 176.7
3158645629883, "max": 259.1421604156494}, "finalize.time": {"sum": 275.042533
8745117, "count": 1, "min": 275.0425338745117, "max": 275.0425338745117}, "se
tuptime": {"sum": 20.176410675048828, "count": 1, "min": 20.176410675048828,
 "max": 20.176410675048828}, "totaltime": {"sum": 3528.4738540649414, "coun
t": 1, "min": 3528.4738540649414, "max": 3528.4738540649414}}}

2021-10-12 13:44:59 Uploading - Uploading generated training model
2021-10-12 13:45:29 Completed - Training job completed
Training seconds: 80
Billable seconds: 80
CPU times: user 587 ms, sys: 76.8 ms, total: 664 ms
Wall time: 4min 43s

In [18]:

```
linear_predictor = linear.deploy(initial_instance_count=1, instance_type='ml.t2.large')
```

```
---------------!
```

In [20]:

```
from sagemaker.predictor import csv_serializer, json_deserializer

linear_predictor.serializer = sagemaker.serializers.CSVSerializer()
linear_predictor.deserializer = sagemaker.deserializers.JSONDeserializer()
```

In [58]:

```
result = linear_predictor.predict(np.array(x_valid))
print(result)
```

{'predictions': [{'score': 0.42480480670928955, 'predicted_label': 0}, {'scor
e': 0.6581493020057678, 'predicted_label': 1}, {'score': 0.5354257822036743,
'predicted_label': 1}, {'score': 0.02998901531100273, 'predicted_label': 0},
{'score': 0.2599537670612335, 'predicted_label': 0}, {'score': 0.084231123328
20892, 'predicted_label': 0}, {'score': 0.45470723509788513, 'predicted_labe
l': 0}, {'score': 0.30146244168281555, 'predicted_label': 0}, {'score': 0.175
78236758708954, 'predicted_label': 0}, {'score': 0.02113206684589386, 'predic
ted_label': 0}, {'score': 0.7052819728851318, 'predicted_label': 1}, {'scor
e': 0.7143887877464294, 'predicted_label': 1}, {'score': 0.7613704800605774,
'predicted_label': 1}, {'score': 0.7692782282829285, 'predicted_label': 1},
{'score': 0.06663920730352402, 'predicted_label': 0}, {'score': 0.02774215862
1549606, 'predicted_label': 0}, {'score': 0.7563450932502747, 'predicted_labe
l': 1}, {'score': 0.2242158204317093, 'predicted_label': 0}, {'score': 0.2686
00195646286, 'predicted_label': 0}, {'score': 0.22995813190937042, 'predicted
_label': 0}, {'score': 0.1084669679403305, 'predicted_label': 0}, {'score':
0.4516526460647583, 'predicted_label': 0}, {'score': 0.140010267496109, 'pred
icted_label': 0}, {'score': 0.787925660610199, 'predicted_label': 1}, {'scor
e': 0.7605970501899719, 'predicted_label': 1}, {'score': 0.653394341468811,
'predicted_label': 1}, {'score': 0.786648154258728, 'predicted_label': 1},
{'score': 0.14153550565242767, 'predicted_label': 0}, {'score': 0.95381039381
02722, 'predicted_label': 1}, {'score': 0.8926922678947449, 'predicted_labe
l': 1}, {'score': 0.03187188878655434, 'predicted_label': 0}, {'score': 0.166
2725806236267, 'predicted_label': 0}, {'score': 0.051000211387872696, 'predic
ted_label': 0}, {'score': 0.03447738289833069, 'predicted_label': 0}, {'scor
e': 0.48487672209739685, 'predicted_label': 1}, {'score': 0.1518832892179489,
'predicted_label': 0}, {'score': 0.3326621949672699, 'predicted_label': 0},
{'score': 0.48288607597351074, 'predicted_label': 1}, {'score': 0.74100947380
06592, 'predicted_label': 1}, {'score': 0.38007065653800964, 'predicted_labe
l': 0}, {'score': 0.20180147886276245, 'predicted_label': 0}, {'score': 0.048
54688048362732, 'predicted_label': 0}, {'score': 0.10551843047142029, 'predic
ted_label': 0}, {'score': 0.6177523136138916, 'predicted_label': 1}, {'scor
e': 0.3383467197418213, 'predicted_label': 0}, {'score': 0.6365821361541748,
'predicted_label': 1}, {'score': 0.0982552319765091, 'predicted_label': 0},
{'score': 0.16908292472362518, 'predicted_label': 0}, {'score': 0.85679215192
7948, 'predicted_label': 1}, {'score': 0.06342484056949615, 'predicted_labe
l': 0}, {'score': 0.6360915899276733, 'predicted_label': 1}, {'score': 0.7242
143750190735, 'predicted_label': 1}, {'score': 0.17997826635837555, 'predicte
d_label': 0}, {'score': 0.08871788531541824, 'predicted_label': 0}, {'score':
0.1058778464794159, 'predicted_label': 0}, {'score': 0.2491646111011505, 'pre
dicted_label': 0}, {'score': 0.09486322104930878, 'predicted_label': 0}, {'sc
ore': 0.5476478934288025, 'predicted_label': 1}, {'score': 0.1766867190599441
5, 'predicted_label': 0}, {'score': 0.03877781704068184, 'predicted_label':
0}, {'score': 0.8521596789360046, 'predicted_label': 1}, {'score': 0.12476874
887943268, 'predicted_label': 0}, {'score': 0.2891680896282196, 'predicted_la
bel': 0}, {'score': 0.5708218216896057, 'predicted_label': 1}, {'score': 0.34
348490834236145, 'predicted_label': 0}, {'score': 0.3374444246292114, 'predic
ted_label': 0}, {'score': 0.8717955946922302, 'predicted_label': 1}, {'scor
e': 0.21385139226913452, 'predicted_label': 0}, {'score': 0.8971049785614014,
'predicted_label': 1}, {'score': 0.404753714799881, 'predicted_label': 0},
{'score': 0.10340692847967148, 'predicted_label': 0}, {'score': 0.74925190210
34241, 'predicted_label': 1}, {'score': 0.5007069110870361, 'predicted_labe
l': 1}, {'score': 0.3830896317958832, 'predicted_label': 0}, {'score': 0.8666
815161705017, 'predicted_label': 1}, {'score': 0.12740565836429596, 'predicte
d_label': 0}, {'score': 0.29697948694229126, 'predicted_label': 0}, {'score':
0.018706919625401497, 'predicted_label': 0}, {'score': 0.5174866318702698, 'p
redicted_label': 1}, {'score': 0.35125479102134705, 'predicted_label': 0},
{'score': 0.8502010107040405, 'predicted_label': 1}, {'score': 0.005850044079

1249275, 'predicted_label': 0}, {'score': 0.5032865405082703, 'predicted_labe
l': 1}, {'score': 0.32886359095573425, 'predicted_label': 0}, {'score': 0.587
8629684448242, 'predicted_label': 1}, {'score': 0.9674394130706787, 'predicte
d_label': 1}, {'score': 0.1132545992732048, 'predicted_label': 0}, {'score':
0.27684926986694336, 'predicted_label': 0}, {'score': 0.2588479816913605, 'pr
edicted_label': 0}, {'score': 0.6709524393081665, 'predicted_label': 1}, {'sc
ore': 0.08120451867580414, 'predicted_label': 0}, {'score': 0.119277894496917
72, 'predicted_label': 0}, {'score': 0.2236638218164444, 'predicted_label':
0}, {'score': 0.4905351996421814, 'predicted_label': 1}, {'score': 0.61001068
35365295, 'predicted_label': 1}, {'score': 0.8814276456832886, 'predicted_lab
el': 1}, {'score': 0.3282400071620941, 'predicted_label': 0}, {'score': 0.309
6885681152344, 'predicted_label': 0}, {'score': 0.08296120166778564, 'predict
ed_label': 0}, {'score': 0.18929165601730347, 'predicted_label': 0}, {'scor
e': 0.5367108583450317, 'predicted_label': 1}, {'score': 0.9079782962799072,
'predicted_label': 1}, {'score': 0.2389388084411621, 'predicted_label': 0},
{'score': 0.4467237889766693, 'predicted_label': 0}, {'score': 0.316262036561
96594, 'predicted_label': 0}, {'score': 0.7957831025123596, 'predicted_labe
l': 1}, {'score': 0.24224606156349182, 'predicted_label': 0}, {'score': 0.769
3429589271545, 'predicted_label': 1}, {'score': 0.2467987984418869, 'predicte
d_label': 0}, {'score': 0.39433056116104126, 'predicted_label': 0}, {'score':
0.8405275940895081, 'predicted_label': 1}, {'score': 0.10653848946094513, 'pr
edicted_label': 0}, {'score': 0.17153514921665192, 'predicted_label': 0}, {'s
core': 0.764477014541626, 'predicted_label': 1}, {'score': 0.0410737097263336
2, 'predicted_label': 0}, {'score': 0.44331300258636475, 'predicted_label':
0}, {'score': 0.4013848602771759, 'predicted_label': 0}, {'score': 0.75264483
69026184, 'predicted_label': 1}, {'score': 0.31918686628341675, 'predicted_la
bel': 0}, {'score': 0.5754538178443909, 'predicted_label': 1}, {'score': 0.40
71899950504303, 'predicted_label': 0}, {'score': 0.3200340270996094, 'predict
ed_label': 0}, {'score': 0.025076974183321, 'predicted_label': 0}, {'score':
0.189071923494339, 'predicted_label': 0}, {'score': 0.15775194764137268, 'pre
dicted_label': 0}, {'score': 0.07133544236421585, 'predicted_label': 0}, {'sc
ore': 0.4349459707736969, 'predicted_label': 0}, {'score': 0.0845164731144905
1, 'predicted_label': 0}, {'score': 0.2910936176776886, 'predicted_label':
0}, {'score': 0.11577177792787552, 'predicted_label': 0}, {'score': 0.0539338
1789326668, 'predicted_label': 0}, {'score': 0.09250804781913757, 'predicted_
label': 0}, {'score': 0.33823058009147644, 'predicted_label': 0}, {'score':
0.08044451475143433, 'predicted_label': 0}, {'score': 0.23410846292972565, 'p
redicted_label': 0}, {'score': 0.14129287004470825, 'predicted_label': 0},
{'score': 0.6853272318840027, 'predicted_label': 1}, {'score': 0.235459014773
36884, 'predicted_label': 0}, {'score': 0.9690595865249634, 'predicted_labe
l': 1}, {'score': 0.31236302852630615, 'predicted_label': 0}, {'score': 0.042
301733046770096, 'predicted_label': 0}, {'score': 0.1072298213839531, 'predic
ted_label': 0}, {'score': 0.1219564825296402, 'predicted_label': 0}, {'scor
e': 0.754566490650177, 'predicted_label': 1}, {'score': 0.9124706983566284,
'predicted_label': 1}, {'score': 0.8398250937461853, 'predicted_label': 1},
{'score': 0.3136586844921112, 'predicted_label': 0}, {'score': 0.184836834669
11316, 'predicted_label': 0}, {'score': 0.3647347092628479, 'predicted_labe
l': 0}, {'score': 0.02846803516149521, 'predicted_label': 0}, {'score': 0.706
088125705719, 'predicted_label': 1}, {'score': 0.003961784765124321, 'predict
ed_label': 0}, {'score': 0.624333381652832, 'predicted_label': 1}, {'score':
0.1231444701552391, 'predicted_label': 0}]}

In [65]:

```
predictions = np.array([res['score'] for res in result['predictions']])
pred_class=np.array([res['predicted_label'] for res in result['predictions']])
```

In [60]:

```
predictions
```

Out[60]:

```
array([0.42480481, 0.6581493 , 0.53542578, 0.02998902, 0.25995377,
       0.08423112, 0.45470724, 0.30146244, 0.17578237, 0.02113207,
       0.70528197, 0.71438879, 0.76137048, 0.76927823, 0.06663921,
       0.02774216, 0.75634509, 0.22421582, 0.2686002 , 0.22995813,
       0.10846697, 0.45165265, 0.14001027, 0.78792566, 0.76059705,
       0.65339434, 0.78664815, 0.14153551, 0.95381039, 0.89269227,
       0.03187189, 0.16627258, 0.05100021, 0.03447738, 0.48487672,
       0.15188329, 0.33266219, 0.48288608, 0.74100947, 0.38007066,
       0.20180148, 0.04854688, 0.10551843, 0.61775231, 0.33834672,
       0.63658214, 0.09825523, 0.16908292, 0.85679215, 0.06342484,
       0.63609159, 0.72421438, 0.17997827, 0.08871789, 0.10587785,
       0.24916461, 0.09486322, 0.54764789, 0.17668672, 0.03877782,
       0.85215968, 0.12476875, 0.28916809, 0.57082182, 0.34348491,
       0.33744442, 0.87179559, 0.21385139, 0.89710498, 0.40475371,
       0.10340693, 0.7492519 , 0.50070691, 0.38308963, 0.86668152,
       0.12740566, 0.29697949, 0.01870692, 0.51748663, 0.35125479,
       0.85020101, 0.00585004, 0.50328654, 0.32886359, 0.58786297,
       0.96743941, 0.1132546 , 0.27684927, 0.25884798, 0.67095244,
       0.08120452, 0.11927789, 0.22366382, 0.4905352 , 0.61001068,
       0.88142765, 0.32824001, 0.30968857, 0.0829612 , 0.18929166,
       0.53671086, 0.9079783 , 0.23893881, 0.44672379, 0.31626204,
       0.7957831 , 0.24224606, 0.76934296, 0.2467988 , 0.39433056,
       0.84052759, 0.10653849, 0.17153515, 0.76447701, 0.04107371,
       0.443313  , 0.40138486, 0.75264484, 0.31918687, 0.57545382,
       0.40719   , 0.32003403, 0.02507697, 0.18907192, 0.15775195,
       0.07133544, 0.43494597, 0.08451647, 0.29109362, 0.11577178,
       0.05393382, 0.09250805, 0.33823058, 0.08044451, 0.23410846,
       0.14129287, 0.68532723, 0.23545901, 0.96905959, 0.31236303,
       0.04230173, 0.10722982, 0.12195648, 0.75456649, 0.9124707 ,
       0.83982509, 0.31365868, 0.18483683, 0.36473471, 0.02846804,
       0.70608813, 0.00396178, 0.62433338, 0.12314447])
```

In [62]:

```
from sklearn import metrics
Y_valid=np.array(y_valid)
np.sqrt(metrics.mean_squared_error(Y_valid, predictions))
```

Out[62]:

```
0.4070087106172575
```

In [55]:

```
prediction_accuracy =np.mean ((valid_pred_class==y_valid)) * 100
```

In [67]:

```
from sklearn.metrics import classification_report
print('Linear_learner','\n',classification_report(y_valid,pred_class))
```

```
Linear_learner
               precision    recall  f1-score   support

           0       0.78      0.81      0.79        98
           1       0.64      0.61      0.62        56

    accuracy                           0.73       154
   macro avg       0.71      0.71      0.71       154
weighted avg       0.73      0.73      0.73       154
```

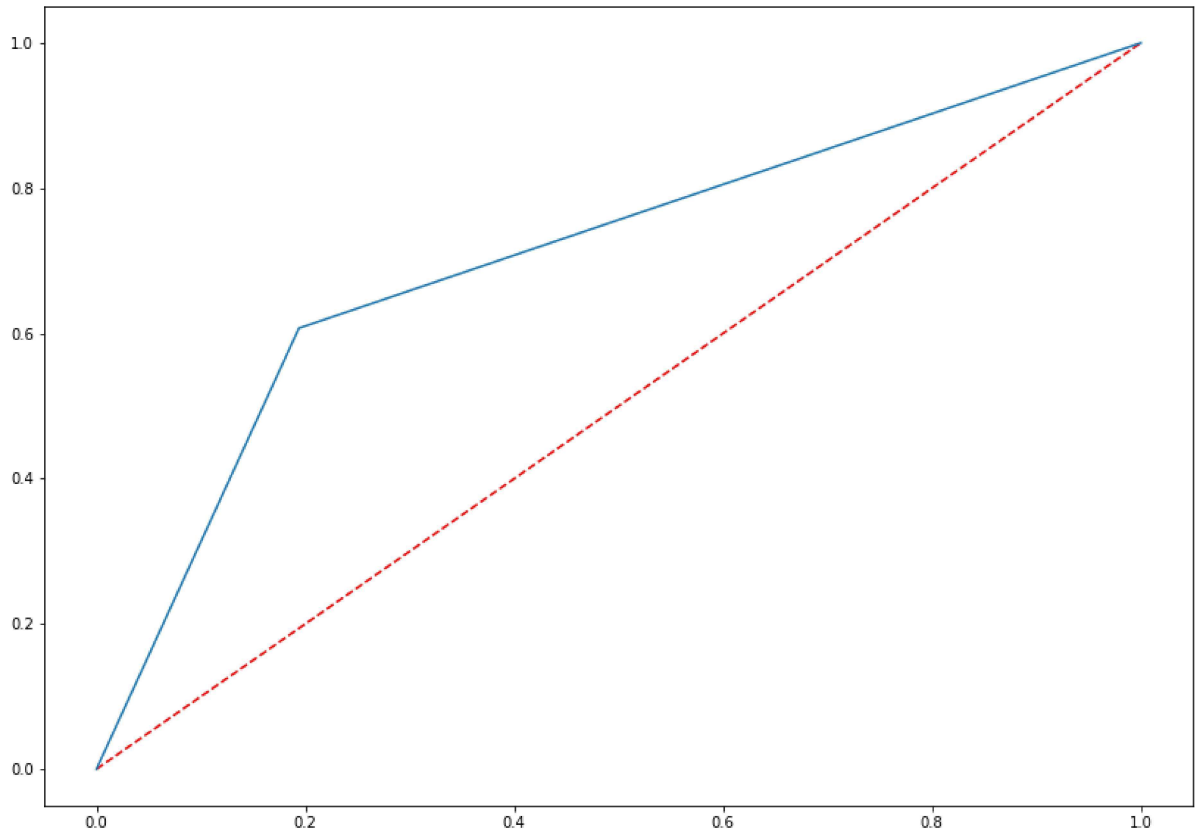
In [74]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_validate

fig = plt.figure(figsize=(14,10))
plt.plot([0, 1], [0, 1],'r--')
fpr, tpr, thresh = metrics.roc_curve(Y_valid,pred_class )
plt.plot(fpr, tpr, label=f'Linear learner, AUC = {str(round(auclg,3))}')
```

Out[74]:

```
[<matplotlib.lines.Line2D at 0x7fab5426e750>]
```



In [76]:

```
auclr= roc_auc_score(Y_valid, pred_class)
auclr
```

Out[76]:

```
0.7066326530612245
```

In [103]:

```
from sklearn.metrics import log_loss

loss = log_loss(Y_valid,predictions)
loss
```

Out[103]:

```
0.5049192309275556
```

In [106]:

```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(Y_valid, predictions)
f1 = f1_score(Y_valid, pred_class)
auc = auc(recall, precision)
```

In [107]:

```
auc
```

Out[107]:

```
0.6928412897077163
```

In [108]:

```
f1
```

Out[108]:

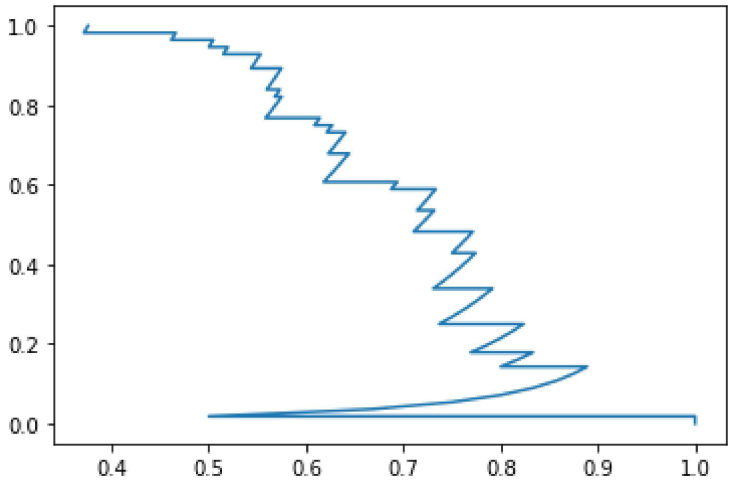
```
0.6238532110091742
```

In [109]:

```
plt.plot(precision,recall)
```

Out[109]:

```
[<matplotlib.lines.Line2D at 0x7fab542b3810>]
```




In [ ]: