MACHINE LEARNING
WITH SPARK

NIHARIKA REDDY

MOTIVATION

- Prediction of heart disease based in patients.
- 13 features- Age, Sex, Resting BP, Cholesterol, FBS, Rest ECG, Maximum heart Rate, etc.
- Getting familiar with Spark functionality

Code Snippets

Importing dataset from s3 bucket

```
bucket='odl-spark19spds6003-001'
data_key = 'nb7ug/heart.csv' # Where the file is within the bucket
data_location = 's3://{}/{}'.format(bucket, data_key)
dataset = pd.read_csv(data_location)
```

```
dataset.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Converting dataframe to parquet

```
df = sqlc.createDataFrame(dataset)
parquetPath = '/home/ec2-user/SageMaker/Spark19SpDS6003-001/nb7ug/parquet-data-heart'
df.write.parquet(parquetPath)
```

MACHINE LEARNING

Preparing a well defined dataset for machine learning

```
# Data pre-processing before building a model
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, VectorAssembler


assembler_features = VectorAssembler(inputCols=['age','sex','cp','trestbps','chol','fbs','restecg', 'thalach','exang','
stages = [assembler_features]
label_stringIdx = StringIndexer(inputCol="target", outputCol="label")
stages += [label_stringIdx]
pipeline = Pipeline(stages=stages)
```

```
#Split into training and validation sets
allData = pipeline.fit(df).transform(df)
allData.cache()
trainingData, testData = allData.randomSplit([0.8,0.2], seed=0) # need to ensure same split for each time
print("Distribution of Positives and Negatives in trainingData is: ", trainingData.groupBy("label").count().take(2))
```

```
Distribution of Positives and Negatives in trainingData is:  [Row(label=0.0, count=133), Row(label=1.0, count=117)]
```

Train and prediction

```
#Using random forest with no. of trees = 5
from pyspark.ml.classification import RandomForestClassifier as RF

rf = RF(labelCol='label', featuresCol='features',numTrees=5)
fit = rf.fit(trainingData)
transformed = fit.transform(testData)
```

Visualisation