

# Predict Heart Failure

## *Lab 6 Guide*



---

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright International Business Machines Corporation 2019.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

## Contents

<b>SECTION 1.</b>	<b>PREFACE .....</b>	<b>4</b>
	OVERVIEW.....	4
	OBJECTIVES .....	6
	TOOLS.....	6
	FLOW.....	7
	PREREQUISITES .....	7
<b>SECTION 2.</b>	<b>CREATE THE NOTEBOOK.....</b>	<b>8</b>
	RUN THE NOTEBOOK .....	10
	ANALYZE THE RESULTS .....	10
<b>SECTION 3.</b>	<b>LOAD AND EXPLORE DATA .....</b>	<b>11</b>
	INTERACTIVE VISUALIZATIONS W/PIXIEDUST.....	15
	<b>A. SIMPLE VISUALIZATION USING BAR CHARTS .....</b>	<b>16</b>
	CREATE AN APACHE® SPARK MACHINE LEARNING MODEL.....	16
	<b>A. PREPARE DATA .....</b>	<b>16</b>
	<b>B. CREATE PIPELINE AND TRAIN A MODEL.....</b>	<b>16</b>
	<b>C. COMPUTE METRICS AS A FUNCTION OF THRESHOLD.....</b>	<b>21</b>
	<b>D. RUN PARAMETER SWEEPS.....</b>	<b>23</b>

---

## Section 1. Preface

### Overview

According to the [Mayo Clinic](#) publication, heart failure, sometimes known as congestive heart failure, occurs when your heart muscle doesn't pump blood as well as it should. Certain conditions, such as narrowed arteries in your heart (coronary artery disease) or high blood pressure, gradually leave your heart too weak or stiff to fill and pump efficiently.

Not all conditions that lead to heart failure can be reversed, but treatments can improve the signs and

One way to prevent heart failure is to prevent and control conditions that cause heart failure, such as coronary artery disease, high blood pressure, diabetes or obesity.

Heart failure often develops after other conditions have damaged or weakened your heart. However, the heart doesn't need to be weakened to cause heart failure. It can also occur if the heart becomes too stiff.

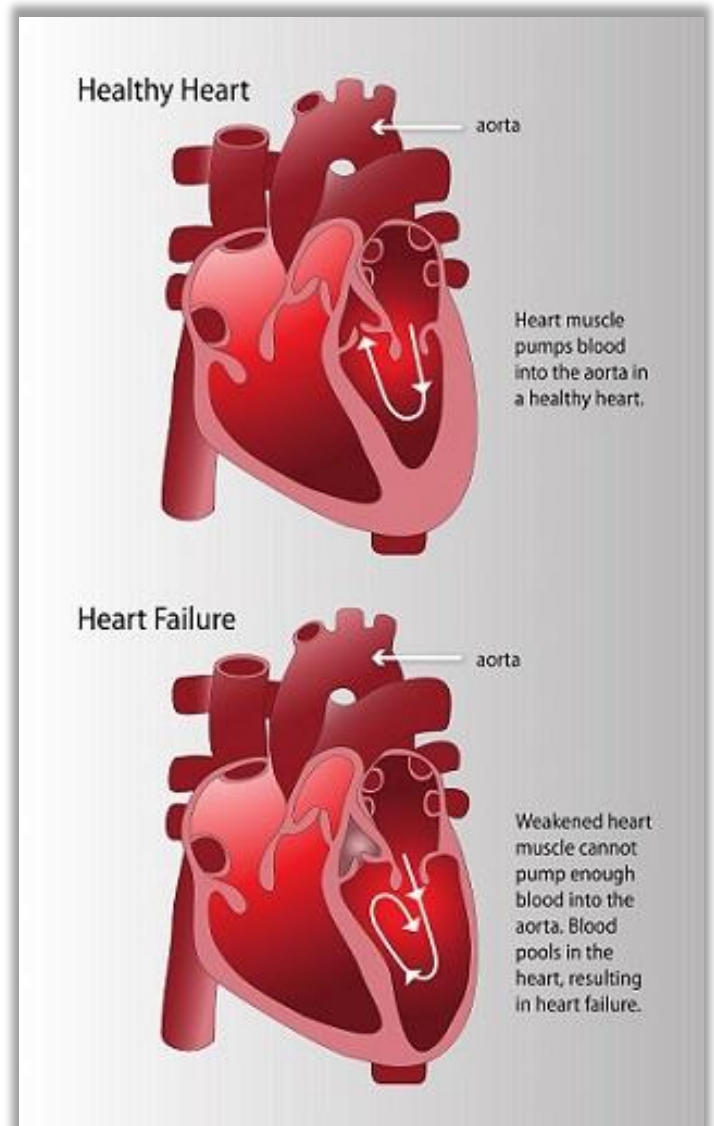
In heart failure, the main pumping chambers of your heart (the ventricles) may become stiff and not fill properly between beats. In some cases of heart failure, your heart muscle may become damaged and weakened, and the ventricles stretch (dilate) to the point that the heart can't pump blood efficiently throughout your body.

Over time, the heart can no longer keep up with the normal demands placed on it to pump blood to the rest of your body.

A single risk factor may be enough to cause heart failure, but a combination of factors also increases your risk.

Risk factors include:

- **High blood pressure.** Your heart works harder than it has to if your blood pressure is high.
- **Coronary artery disease.** Narrowed arteries may limit your heart's supply of oxygen-rich blood, resulting in weakened heart muscle.
- **Heart attack.** A heart attack is a form of coronary disease that occurs suddenly. Damage to your heart muscle from a heart attack may mean your heart can no longer pump as well as it should.
- **Diabetes.** Having diabetes increases your risk of high blood pressure and coronary artery disease.
- **Sleep apnea.** The inability to breathe properly while you sleep at night results in low blood oxygen levels and increased risk of abnormal heart rhythms. Both of these problems can weaken the heart.
- **Congenital heart defects.** Some people who develop heart failure were born with structural heart defects.
- **Valvular heart disease.** People with valvular heart disease have a higher risk of heart failure.
- **Viruses.** A viral infection may have damaged your heart muscle.
- **Alcohol use.** Drinking too much alcohol can weaken heart muscle and lead to heart failure.
- **Tobacco use.** Using tobacco can increase your risk of heart failure.
- **Obesity.** People who are obese have a higher risk of developing heart failure.
- **Irregular heartbeats.** These abnormal rhythms, especially if they are very frequent and fast, can weaken the heart muscle and cause heart failure.



This notebook contains steps and code to create a predictive model to predict heart failure and then deploy that model to Watson Machine Learning so it can be used in an application.

This model was modified by Dr. Jeremy Nilmeire, Developer Advocate at IBM and originally adopted from Justin McCoy's notebook, which can be found at <https://github.com/IBM/predictive-model-on-watson-ml>

## Objectives

The learning goals of this notebook are:

- Either load the patientDataV6 file into the Object Storage Service linked to your Watson Studio project or import the Jupyter Notebook into your project.
- Create an Apache® Spark machine learning model
- Train and evaluate a model
- Persist a model in a Watson Machine Learning repository

## Tools



Watson Studio



Jupyter Notebook

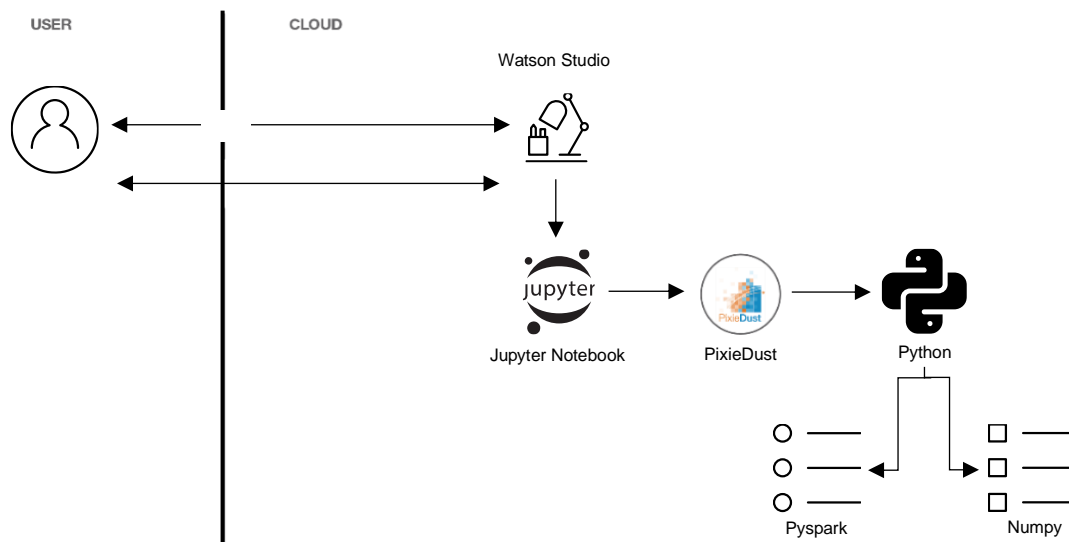


PixieDust



Python (Pyspark and Numpy)

## Flow



1. The User will create a Watson Studio Service.
2. From Watson Studio connect to Jupyter Notebook.
3. Utilize PixieDust, an open-source visualization program.
4. Manipulate Python code using Python Libraries, Pyspark and Numpy.

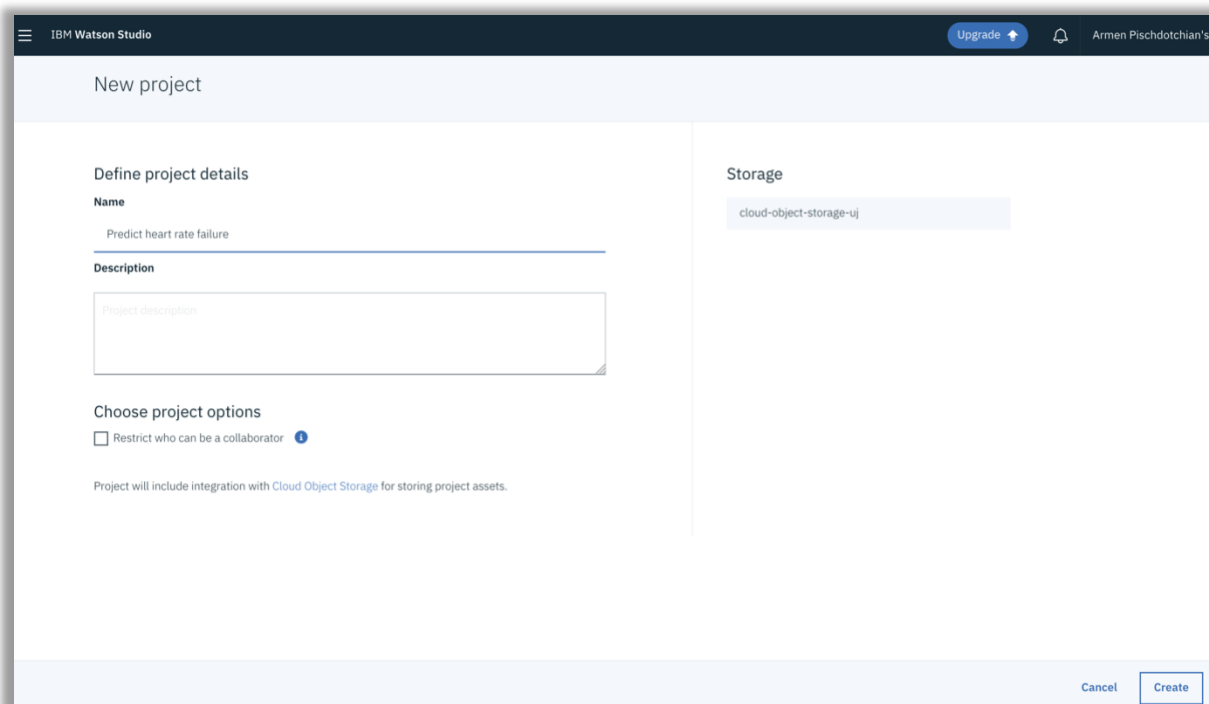
## Prerequisites

This lab assumes that you have completed all prior labs and that you have an active IBM Cloud Account.

## Section 2. Create the Notebook

By now you have already registered with IBM Cloud and applied your promo code. Let's begin our journey:

1. Login into IBM Cloud: <https://cloud.ibm.com>
2. Remove the **Label:lite** filter.
3. Click the **Catalog** tab.
4. Search for the **Watson Studio** service and click that tile.
5. Click **Create**.
6. Click the **Get Started** button.
7. Click **Get started** again.
8. Click **Create a project**.
9. Select the **Create an empty project** tile.
10. Specify a name; for example: Predict heart failure rate



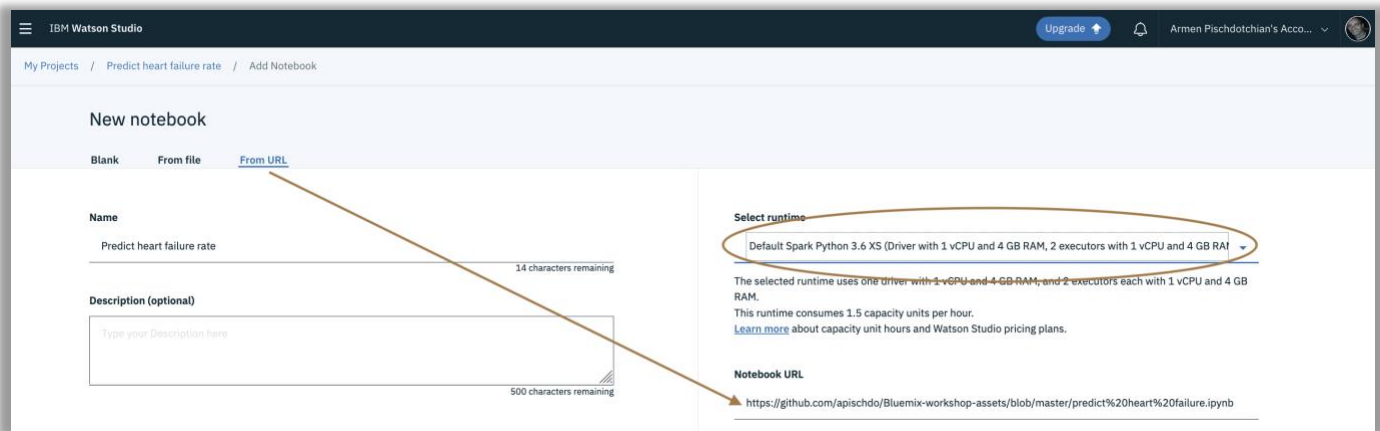
The screenshot shows the 'New project' form in IBM Watson Studio. The form is divided into two main sections: 'Define project details' and 'Storage'. In the 'Define project details' section, the 'Name' field is filled with 'Predict heart rate failure'. The 'Description' field is empty. In the 'Storage' section, the 'cloud-object-storage-uj' option is selected. At the bottom of the form, there is a 'Choose project options' section with a checkbox labeled 'Restrict who can be a collaborator' which is currently unchecked. Below this, a note states: 'Project will include integration with Cloud Object Storage for storing project assets.' At the bottom right of the form, there are two buttons: 'Cancel' and 'Create'.

11. Click **Create**
12. Click **Add to project**.

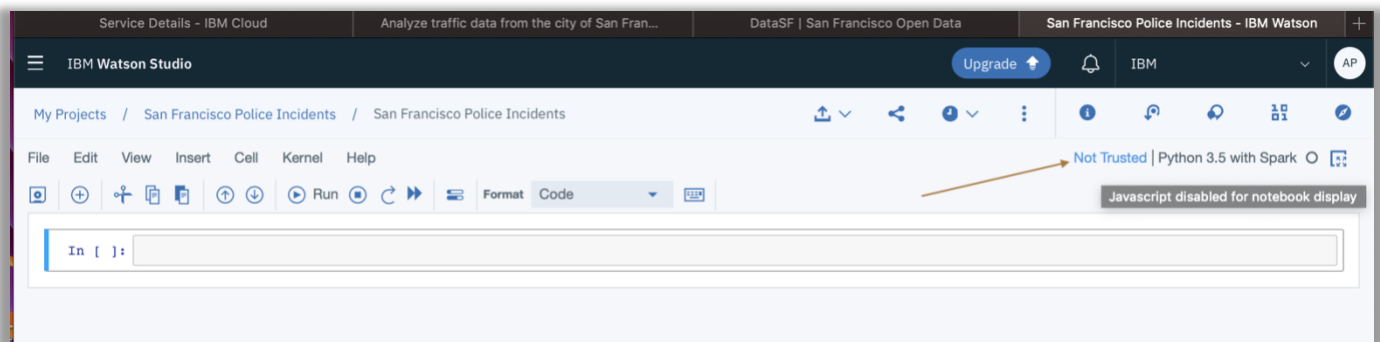


13. Click the **NOTEBOOK** tile.
14. Specify a name.
15. From the **Select runtime** drop down list, select the *Spark* runtime.  
**Default Spark Python 3.6 XS**
16. Click **From URL** and copy and paste the following URL:

<https://ibm.box.com/s/w4hm2a9je4k0y629j5dkz6yvxdz8ecg>



17. Click **Create Notebook**.
18. Change the Kernel to **Trusted** (top right corner of the notebook).



## Run the notebook

When a notebook is executed, what is actually happening is that each code cell in the notebook is executed, in order, from top to bottom.

Each code cell is selectable and is preceded by a tag in the left margin. The tag format is In [x]:. Depending on the state of the notebook, the x can be:

- A blank, this indicates that the cell has never been executed.
- A number, this number represents the relative order this code step was executed.
- A \*, this indicates that the cell is currently executing.

There are several ways to execute the code cells in your notebook:

- One cell at a time.
  - Select the cell, and then press the Play button in the toolbar.
- Batch mode, in sequential order.
  - From the Cell menu bar, there are several options available. For example, you can Run All cells in your notebook, or you can Run All Below, that will start executing from the first cell under the currently selected cell, and then continue executing all cells that follow.
- At a scheduled time.
  - Press the Schedule button located in the top right section of your notebook panel. Here you can schedule your notebook to be executed once at some future time, or repeatedly at your specified interval.

## Analyze the Results

After running each cell of the notebook, the results will display.

## Section 3. Load and Explore Data

In this section you will load the data as an Apache® Spark DataFrame and perform a basic exploration.

Load the data to the Spark DataFrame from your associated Object Storage instance. Once Data is loaded the rest of this exercise will just be following along an interpreting the data presented.

**In [2]:**

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
spark
```

**Out[2]:**

**SparkSession - in-memory**

**SparkContext**

Version

v2.3.0

Master

local[\*]

AppName

pyspark-shell

**In [3]:**

```
df_data = spark.read\
    .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
    .option('header', 'true')\
    .option('inferSchema', 'true')\
    .load('patientdataV6.csv')
df_data.take(5)
```

**Out[3]:**

[Row(AVGHEARTBEATSPERMIN=93, PALPITATIONSPERDAY=22, CHOLESTEROL=163, BMI=25, HEARTFAILURE='N', AGE=49, SEX='F', FAMILYHISTORY='N', SMOKERLAST5YRS='N', EXERCISEMINPERWEEK=110),

Row(AVGHEARTBEATSPERMIN=108, PALPITATIONSPERDAY=22, CHOLESTEROL=181, BMI=24, HEARTFAILURE='N', AGE=32, SEX='F', FAMILYHISTORY='N', SMOKERLAST5YRS='N', EXERCISEMINPERWEEK=192),

Row(AVGHEARTBEATSPERMIN=86, PALPITATIONSPERDAY=0, CHOLESTEROL=239, BMI=20, HEARTFAILURE='N', AGE=60, SEX='F', FAMILYHISTORY='N', SMOKERLAST5YRS='N', EXERCISEMINPERWEEK=121),

```
Row(AVGHEARTBEATSPERMIN=80, PALPITATIONSPERDAY=36, CHOLESTEROL=164, BMI=31,
HEARTFAILURE='Y', AGE=45, SEX='F', FAMILYHISTORY='Y', SMOKERLAST5YRS='N',
EXERCISEMINPERWEEK=141),
```

```
Row(AVGHEARTBEATSPERMIN=66, PALPITATIONSPERDAY=36, CHOLESTEROL=185, BMI=23,
HEARTFAILURE='N', AGE=39, SEX='F', FAMILYHISTORY='N', SMOKERLAST5YRS='N',
EXERCISEMINPERWEEK=63)]
```

Explore the loaded data by using the following Apache® Spark DataFrame methods:

- print schema
- print top ten records
- count all records

In [4]:

```
df_data.printSchema()
root
 |-- AVGHEARTBEATSPERMIN: integer (nullable = true)
 |-- PALPITATIONSPERDAY: integer (nullable = true)
 |-- CHOLESTEROL: integer (nullable = true)
 |-- BMI: integer (nullable = true)
 |-- HEARTFAILURE: string (nullable = true)
 |-- AGE: integer (nullable = true)
 |-- SEX: string (nullable = true)
 |-- FAMILYHISTORY: string (nullable = true)
 |-- SMOKERLAST5YRS: string (nullable = true)
 |-- EXERCISEMINPERWEEK: integer (nullable = true)
```

In [5]:

```
df_data.show()
+-----+-----+-----+-----+-----+-----+-----+-----+
|AVGHEARTBEATSPERMIN|PALPITATIONSPERDAY|CHOLESTEROL|BMI|HEARTFAILURE|AGE|SEX|FAMILYHISTORY|SMOKERLAST5YRS|EXERCISEMINPERWEEK|
+-----+-----+-----+-----+-----+-----+-----+-----+
|          93|          22|          163| 25|          N| 49|  F|          N|          |
N|          110|          |          |  |          |  |          |          |
|          108|          22|          181| 24|          N| 32|  F|          N|          |
N|          192|          |          |  |          |  |          |          |
|          86|          0|          239| 20|          N| 60|  F|          N|          |
N|          121|          |          |  |          |  |          |          |
|          80|          36|          164| 31|          Y| 45|  F|          Y|          |
N|          141|          |          |  |          |  |          |          |
```

	66	36	185	23	N	39	F	N
N	63							
	125	27	201	31	N	47	M	N
N	13							
	83	27	169	20	N	71	F	Y
N	124							
	107	31	199	32	N	55	F	N
N	22							
	92	28	174	22	N	44	F	N
N	107							
	84	12	206	25	N	50	M	N
N	199							
	60	1	194	28	N	71	M	N
N	27							
	134	7	228	34	Y	63	F	Y
N	92							
	103	0	237	24	N	64	F	Y
N	34							
	101	39	157	20	N	49	M	N
N	33							
	92	2	169	26	N	36	M	N
N	217							
	80	27	234	27	N	50	M	N
N	28							
	82	14	155	30	N	70	F	N
N	207							
	63	9	204	26	N	42	M	N
N	88							
	83	12	209	29	N	38	M	Y
N	220							
	80	37	157	20	N	48	M	N
N	54							

```
+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+
```

only showing top 20 rows

In [6]:

```
df_data.select("AVGHEARTBEATSPERMIN",
"PALPITATIONSPERDAY","CHOLESTEROL","BMI","HEARTFAILURE","SEX").toPandas().head()
```

Out[6]:

	AVGHEARTBEATSPERMIN	PALPITATIONSPERDAY	CHOLESTEROL	BMI	HEARTFAILURE	SEX
0	93	22	163	25	N	F
1	108	22	181	24	N	F
2	86	0	239	20	N	F
3	80	36	164	31	Y	F
4	66	36	185	23	N	F

As you can see, the data contains ten fields. The HEARTFAILURE field is the one we would like to predict (label).

In [7]:

```
df_data.describe().show()
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|summary|AVGHEARTBEATSPERMIN|PALPITATIONSPERDAY|          CHOLESTEROL|
BMI|HEARTFAILURE|          AGE|  SEX|FAMILYHISTORY|SMOKERLAST5YRS|EXERCISEMINPERWEEK|
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|  count|          10800|          10800|          10800|          10800|
10800|          10800|10800|          10800|          10800|
|  mean|  87.11509259259259|20.423148148148147|195.08027777777778|  26.35972222222222|
null|49.965185185185184| null|          null|          null|119.72953703703703|
| stddev| 19.744375148984474|12.165320351622993|26.136731865042325|  3.8201472810942136|
null|13.079280962015586| null|          null|          null| 71.14706006382843|
|  min|          28|          48|          0|          150|          20|
N|          28|  F|          N|          N|          0|
|  max|          72|          161|          45|          245|          34|
Y|          72|  M|          Y|          Y|          250|
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

In [8]:

```
df_data.count()
```

Out[8]:

```
10800
```

As you can see, the data set contains 10800 records.

## Interactive Visualizations w/PixieDust

In [9]:

```
# To confirm you have the latest version of PixieDust on your system, run this cell
#!pip install --user pixiedust==1.1.2
!pip install pixiedust
Requirement already satisfied: pixiedust==1.1.2 in /root/.local/lib/python3.5/site-packages (1.1.2)
Requirement already satisfied: lxml in /root/anaconda2/envs/py35/lib/python3.5/site-packages (from pixiedust==1.1.2) (3.7.3)
Requirement already satisfied: mpld3 in /root/.local/lib/python3.5/site-packages (from pixiedust==1.1.2) (0.3)
Requirement already satisfied: geojson in /root/.local/lib/python3.5/site-packages (from pixiedust==1.1.2) (2.4.1)
You are using pip version 10.0.1, however version 19.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Requirement already satisfied: pixiedust in /root/.local/lib/python3.5/site-packages (1.1.2)
Requirement already satisfied: lxml in /root/anaconda2/envs/py35/lib/python3.5/site-packages (from pixiedust) (3.7.3)
Requirement already satisfied: mpld3 in /root/.local/lib/python3.5/site-packages (from pixiedust) (0.3)
Requirement already satisfied: geojson in /root/.local/lib/python3.5/site-packages (from pixiedust) (2.4.1)
You are using pip version 10.0.1, however version 19.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
If indicated by the installer, restart the kernel and rerun the notebook until here and continue with the workshop.
```

In [10]:

```
import pixiedust
Pixiedust database opened successfully
```



Pixiedust version 1.1.2

```
Table SPARK_PACKAGES created successfully
```

```
Warning: You are not running the latest version of PixieDust. Current is 1.1.2, Latest is 1.1.15
```

```
Please copy and run the following command in a new cell to upgrade: !pip install --user --upgrade pixiedust
```

Please restart kernel after upgrading.

## A. Simple visualization using bar charts

With PixieDust `display()`, you can visually explore the loaded data using built-in charts, such as, bar charts, line charts, scatter plots, or maps. To explore a data set: choose the desired chart type from the drop down, configure chart options, configure display options.

In [11]:

```
display(df_data)
```

*Hey, there's something awesome here! To see it, open this notebook outside GitHub, in a viewer like Jupyter*

## Create an Apache® Spark machine learning model

In this section you will learn how to prepare data, create and train an Apache® Spark machine learning model.

### A. Prepare data

In this subsection you will split your data into: train and test data sets.

In [12]:

```
split_data = df_data.randomSplit([0.8, 0.2], 24)
train_data = split_data[0]
test_data = split_data[1]
```

```
print("Number of training records: " + str(train_data.count()))
```

```
print("Number of testing records : " + str(test_data.count()))
```

```
Number of training records: 8637
```

```
Number of testing records : 2163
```

As you can see our data has been successfully split into two data sets:

- The train data set, which is the largest group, is used for training.
- The test data set will be used for model evaluation and is used to test the assumptions of the model.

### B. Create pipeline and train a model

In this section you will create an Apache® Spark machine learning pipeline and then train the model. In the first step you need to import the Apache® Spark machine learning packages that will be needed in the subsequent steps.

A sequence of data processing is called a *data pipeline*. Each step in the pipeline processes the data and passes the result to the next step in the pipeline, this allows you to transform and fit your model with the raw input data.



**In [13]:**

```
from pyspark.ml.feature import StringIndexer, IndexToString, VectorAssembler
from pyspark.ml.classification import RandomForestClassifier, LogisticRegression, NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline, Model
```

In the following step, convert all the string fields to numeric ones by using the StringIndexer transformer.

**In [14]:**

```
stringIndexer_label = StringIndexer(inputCol="HEARTFAILURE", outputCol="label").fit(df_data)
stringIndexer_sex = StringIndexer(inputCol="SEX", outputCol="SEX_IX")
stringIndexer_famhist = StringIndexer(inputCol="FAMILYHISTORY",
outputCol="FAMILYHISTORY_IX")
stringIndexer_smoker = StringIndexer(inputCol="SMOKERLAST5YRS",
outputCol="SMOKERLAST5YRS_IX")
```

Next, define estimators you want to use for classification. We will compare performance of Random Forest, Logistic Regression, and Naive Bayes in the following example.

**In [15]:**

```
rf = RandomForestClassifier(labelCol="label", featuresCol="features")
lr = LogisticRegression()
nb = NaiveBayes(smoothing=1.0)
```

Finally, indexed labels back to original labels.

In [16]:

```
labelConverter = IndexToString(inputCol="prediction", outputCol="predictedLabel",
labels=stringIndexer_label.labels)
```

In [17]:

```
vectorAssembler_features = VectorAssembler(inputCols=
["AVGHEARTBEATSPERMIN", "PALPITATIONSPERDAY",
                                     "CHOLESTEROL", "BMI", "AGE", "SEX_IX", "FAMILYHISTORY_IX",
                                     "SMOKERLAST5YRS_IX", "EXERCISEMINPERWEEK"],
outputCol="features")
```

```
transform_df_pipeline = Pipeline(stages=[stringIndexer_label, stringIndexer_sex,
stringIndexer_famhist, stringIndexer_smoker, vectorAssembler_features])
```

```
transformed_df = transform_df_pipeline.fit(df_data).transform(df_data)
```

```
transformed_df.show()
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
|AVGHEARTBEATSPERMIN|PALPITATIONSPERDAY|CHOLESTEROL|BMI|HEARTFAILURE|AGE|SEX|FAMILYHISTORY|S
MOKERLAST5YRS|EXERCISEMINPERWEEK|label|SEX_IX|FAMILYHISTORY_IX|SMOKERLAST5YRS_IX|
features|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
|          93|          22|          163| 25|          N| 49|  F|          N|
N|          110|  0.0|  1.0|          0.0|          0.0|[93.0,22.0,163.0,...|
|          108|          22|          181| 24|          N| 32|  F|          N|
N|          192|  0.0|  1.0|          0.0|          0.0|[108.0,22.0,181.0...|
|          86|          0|          239| 20|          N| 60|  F|          N|
N|          121|  0.0|  1.0|          0.0|          0.0|[86.0,0.0,239.0,2...|
|          80|          36|          164| 31|          Y| 45|  F|          Y|
N|          141|  1.0|  1.0|          1.0|          0.0|[80.0,36.0,164.0,...|
|          66|          36|          185| 23|          N| 39|  F|          N|
N|          63|  0.0|  1.0|          0.0|          0.0|[66.0,36.0,185.0,...|
|          125|          27|          201| 31|          N| 47|  M|          N|
N|          13|  0.0|  0.0|          0.0|          0.0|[125.0,27.0,201.0...|
|          83|          27|          169| 20|          N| 71|  F|          Y|
N|          124|  0.0|  1.0|          1.0|          0.0|[83.0,27.0,169.0,...|
|          107|          31|          199| 32|          N| 55|  F|          N|
N|          22|  0.0|  1.0|          0.0|          0.0|[107.0,31.0,199.0...|
|          92|          28|          174| 22|          N| 44|  F|          N|
N|          107|  0.0|  1.0|          0.0|          0.0|[92.0,28.0,174.0,...|
|          84|          12|          206| 25|          N| 50|  M|          N|
N|          199|  0.0|  0.0|          0.0|          0.0|[84.0,12.0,206.0,...|
```

	60			1	194	28	N	71	M	N
N	27	0.0	0.0		0.0		0.0	[60.0,1.0,194.0,2...		
	134			7	228	34	Y	63	F	Y
N	92	1.0	1.0		1.0		0.0	[134.0,7.0,228.0,...		
	103			0	237	24	N	64	F	Y
N	34	0.0	1.0		1.0		0.0	[103.0,0.0,237.0,...		
	101			39	157	20	N	49	M	N
N	33	0.0	0.0		0.0		0.0	[101.0,39.0,157.0...		
	92			2	169	26	N	36	M	N
N	217	0.0	0.0		0.0		0.0	[92.0,2.0,169.0,2...		
	80			27	234	27	N	50	M	N
N	28	0.0	0.0		0.0		0.0	[80.0,27.0,234.0,...		
	82			14	155	30	N	70	F	N
N	207	0.0	1.0		0.0		0.0	[82.0,14.0,155.0,...		
	63			9	204	26	N	42	M	N
N	88	0.0	0.0		0.0		0.0	[63.0,9.0,204.0,2...		
	83			12	209	29	N	38	M	Y
N	220	0.0	0.0		1.0		0.0	[83.0,12.0,209.0,...		
	80			37	157	20	N	48	M	N
N	54	0.0	0.0		0.0		0.0	[80.0,37.0,157.0,...		

-----+

-----+

-----+

only showing top 20 rows

Let's build our pipelines now. A pipeline consists of transformers and an estimator.

In [18]:

```
pipeline1 = Pipeline(stages=[stringIndexer_label, stringIndexer_sex,
                             stringIndexer_famhist, stringIndexer_smoker,
                             vectorAssembler_features, rf, labelConverter])
```

```
m1Name = "Random Forest"
```

```
pipeline2 = Pipeline(stages=[stringIndexer_label, stringIndexer_sex,
                             stringIndexer_famhist, stringIndexer_smoker,
                             vectorAssembler_features, lr, labelConverter])
```

```
m2Name = "Logistic Regression"
```

```
pipeline3 = Pipeline(stages=[stringIndexer_label, stringIndexer_sex,
                             stringIndexer_famhist, stringIndexer_smoker,
                             vectorAssembler_features, nb, labelConverter])
```

```
m3Name = "Naive Bayes"
```

Now, you can train your Random Forest model by using the previously defined **pipelines** and **training data**.

In [19]:

```
model1 = pipeline1.fit(train_data)
```

```
model2 = pipeline2.fit(train_data)
```

```
model3 = pipeline3.fit(train_data)
```

Here, we write a function to extract the number of true and false positives and negatives from our dataframes.

In [20]:

```
from pyspark.sql.types import Row
```

```
import numpy as np
```

```
def getCMEntries(threshold):
```

```
    newThresholdDF = spark.sql("select label, p1, prediction as oldPrediction,"
                              " case when p1 > " + str(threshold) + " then 1.0 else 0.0 end as newPrediction"
                              " from inputToThreshold")
    newThresholdDF.registerTempTable("newThreshold")
```

```
    # Here is an SQL query to find true positives
```

```
    tpA = spark.sql("SELECT * FROM newThreshold WHERE label = 1 AND newprediction = 1")
```

```
    # Write an SQL query to find the number of false positives
```

```

### spark.sql("???")
fpA = spark.sql("SELECT * FROM newThreshold WHERE label = 0 AND newprediction = 1")

# Write an SQL query to find the number of false negatives
### spark.sql("???")
fnA = spark.sql("SELECT * FROM newThreshold WHERE label = 1 AND newprediction = 0")

# Write an SQL query to find the number of true negatives
### spark.sql("???")
tnA = spark.sql("SELECT * FROM newThreshold WHERE label = 0 AND newprediction = 0")
return (tpA.count(), fpA.count(), fnA.count(), tnA.count())

```

You can check your model accuracy now. To evaluate the model, use test data.

### C. Compute Metrics as a Function of Threshold

It may be the case that the default threshold of 0.5 for classification is not ideal. Let's explore this possibility, and use some standard metrics to evaluate model fitness.

$TP$ ,  $FP$  are the number of true and false positives,  $FN$ ,  $TN$  are the number of false and true negatives.  $P$  and  $N$  are the total number of ground truth positive and negatives.

True Positive Rate:  $TPR = TP / (FP + FN) = TP / P$

False Positive Rate:  $FP / N$

Matthews Correlation Coefficient:  $MCC = \frac{TP \cdot TN - FP \cdot FN}{[(TP + FP) \cdot (FP + FN) \cdot (FN + FP) \cdot (TN + FN)]^{1/2}}$

The Area Under the Curve (AUC):  $AUC = \int_0^1 TPR d(FPR)$

where the y axis is  $TPR$  and the x axis is  $FPR$ . A plot of  $TPR$  versus  $FPR$  is known as the Receiver Operating Characteristic (ROC).

Recall the trapezoid rule:  $\int_{x_0}^{x_N} f(x) dx = \sum_{i=0}^{N-1} (x_{i+1} - x_i) \cdot \frac{1}{2} [f(x_{i+1}) + f(x_i)]$

where  $TPR = f(x)$  and  $FPR = x$ .

Use this formula to compute the AUC for the ROC, which is a plot of TPR (y axis) vs FPR (x axis). The AUC is a measure of performance across all threshold values.

In [21]:

```
import numpy as np
numBins = 10
thresholds = np.array(range(0, numBins + 1))*1.0/numBins
```

Here, we are writing a function to compute `tp`, `fp`, `fn`, `tn`, etc. for each threshold setting (the default is 0.5). In many cases, a slightly different threshold can perform better. You are asked to fill in the blanks marked with "???".

In [22]:

```
def getModelThresholdStats(model_df, data):
    tp = np.array([i for i in range(0, numBins + 1)])
    fp = np.array([i for i in range(0, numBins + 1)])
    fn = np.array([i for i in range(0, numBins + 1)])
    tn = np.array([i for i in range(0, numBins + 1)])

    #generate dataframe to be used in thresholding:
    predictionsForROC = model_df.transform(data)
    predictionsForROC.registerTempTable("predictions")
    columnsForCM = spark.sql("select probability, prediction, label from predictions")
    extractedProbability = columnsForCM.rdd.map(lambda x: Row(p1 = np
        .asscalar(x[0][1]), prediction=x[1] , label=x[2])).toDF()
    extractedProbability.registerTempTable("inputToThreshold")

    # get the total number of positives and negatives in the predictions dataset:
    ### p = spark.sql("???").count()
    ### n = spark.sql("???").count()
    p = spark.sql("SELECT * from predictions WHERE label = 1").count()
    n = spark.sql("SELECT * from predictions WHERE label = 0").count()

    # We know the number of true positives, etc. at the threshold edges:
    (tp[0],fp[0],fn[0],tn[0]) = (p, n, 0, 0)
    (tp[-1],fp[-1],fn[-1],tn[-1]) = (0, 0, p, n)

    for (i, threshold) in zip(range(0, numBins + 1),thresholds):
        print(i, threshold)
        if (i>0 and i<numBins):
            (tp[i],fp[i],fn[i],tn[i]) = getCMEntries(threshold)
```

```

    print(tp[i],fp[i],fn[i],tn[i])

# coercing to double precision from integers
tp = tp*1.0; fp=fp*1.0; p=p*1.0; n=n*1.0

# calculate the true positive and false positive rate
### tpr = ???
### fpr = ???
tpr = tp/(tp + fn)
fpr = fp/(fp + tn)

# calculate Matthews Correlation Coefficient
### mcc = ???
mcc = (tp * tn - fp * fn)/np.sqrt((tp + fp)*(tp + fn) * (tn + fp) * (tn + fn))

# calculate accuracy as a function of threshold
accThreshold = (tp + tn) / (p + n)

# calculate the area under the curve
auc = - np.array(
    [(fpr[i + 1] - fpr[i]) * 0.5 * (tpr[i + 1] + tpr[i]) for i in range(0,numBins)]
).sum()

return (tpr, fpr, mcc, accThreshold, auc)

```

## D. Run Parameter Sweeps

In [23]:

```

print("getting stats for " + m1Name + ": train")
(tpr1,fpr1,mcc1,acc1,auc1) = getModelThresholdStats(model1, train_data)
print("getting stats for " + m1Name + ": test")
(tpr1Test,fpr1Test,mcc1Test,acc1Test,auc1Test) = getModelThresholdStats(model1, test_data)
print("getting stats for " + m2Name)
(tpr2,fpr2,mcc2,acc2,auc2) = getModelThresholdStats(model2, train_data)
print("getting stats for " + m1Name + ": test")
(tpr2Test,fpr2Test,mcc2Test,acc2Test,auc2Test) = getModelThresholdStats(model2, test_data)
print("getting stats for " + m3Name + ": train")
(tpr3,fpr3,mcc3,acc3,auc3) = getModelThresholdStats(model3, train_data)
print("getting stats for " + m3Name + ": test")

```

```

(tpr3Test,fpr3Test,mcc3Test,acc3Test,auc3Test) = getModelThresholdStats(model3, test_data)
getting stats for Random Forest: train
0 0.0
1427 7210 0 0
1 0.1
970 1383 457 5827
2 0.2
894 893 533 6317
3 0.3
849 697 578 6513
4 0.4
792 545 635 6665
5 0.5
512 166 915 7044
6 0.6
304 52 1123 7158
7 0.7
85 6 1342 7204
8 0.8
2 0 1425 7210
9 0.9
0 0 1427 7210
10 1.0
0 0 1427 7210
getting stats for Random Forest: test
/root/anaconda2/envs/py35/lib/python3.5/site-packages/ipykernel_launcher.py:42:
RuntimeWarning: invalid value encountered in true_divide
0 0.0
361 1802 0 0
1 0.1
254 337 107 1465
2 0.2
232 211 129 1591
3 0.3
225 157 136 1645
4 0.4
201 136 160 1666
5 0.5

```



```
132 51 229 1751
6 0.6
65 18 296 1784
7 0.7
21 2 340 1800
8 0.8
0 0 361 1802
9 0.9
0 0 361 1802
10 1.0
0 0 361 1802
getting stats for Logistic Regression
0 0.0
1427 7210 0 0
1 0.1
1111 3023 316 4187
2 0.2
855 1060 572 6150
3 0.3
740 687 687 6523
4 0.4
595 429 832 6781
5 0.5
425 233 1002 6977
6 0.6
271 95 1156 7115
7 0.7
134 32 1293 7178
8 0.8
34 9 1393 7201
9 0.9
1 1 1426 7209
10 1.0
0 0 1427 7210
getting stats for Random Forest: test
0 0.0
361 1802 0 0
1 0.1
```

```
280 730 81 1072
2 0.2
226 247 135 1555
3 0.3
203 163 158 1639
4 0.4
161 109 200 1693
5 0.5
108 53 253 1749
6 0.6
67 26 294 1776
7 0.7
25 7 336 1795
8 0.8
8 1 353 1801
9 0.9
1 0 360 1802
10 1.0
0 0 361 1802
getting stats for Naive Bayes: train
0 0.0
1427 7210 0 0
1 0.1
876 3314 551 3896
2 0.2
841 3133 586 4077
3 0.3
809 2998 618 4212
4 0.4
785 2905 642 4305
5 0.5
770 2814 657 4396
6 0.6
742 2729 685 4481
7 0.7
715 2639 712 4571
8 0.8
686 2504 741 4706
```

```
9 0.9
643 2335 784 4875
10 1.0
0 0 1427 7210
getting stats for Naive Bayes: test
0 0.0
361 1802 0 0
1 0.1
230 789 131 1013
2 0.2
221 748 140 1054
3 0.3
216 721 145 1081
4 0.4
210 695 151 1107
5 0.5
203 669 158 1133
6 0.6
198 635 163 1167
7 0.7
193 612 168 1190
8 0.8
186 579 175 1223
9 0.9
172 534 189 1268
10 1.0
0 0 361 1802
Now plot your results...
```

In [24]:

```
%matplotlib inline

import matplotlib.pyplot as plt

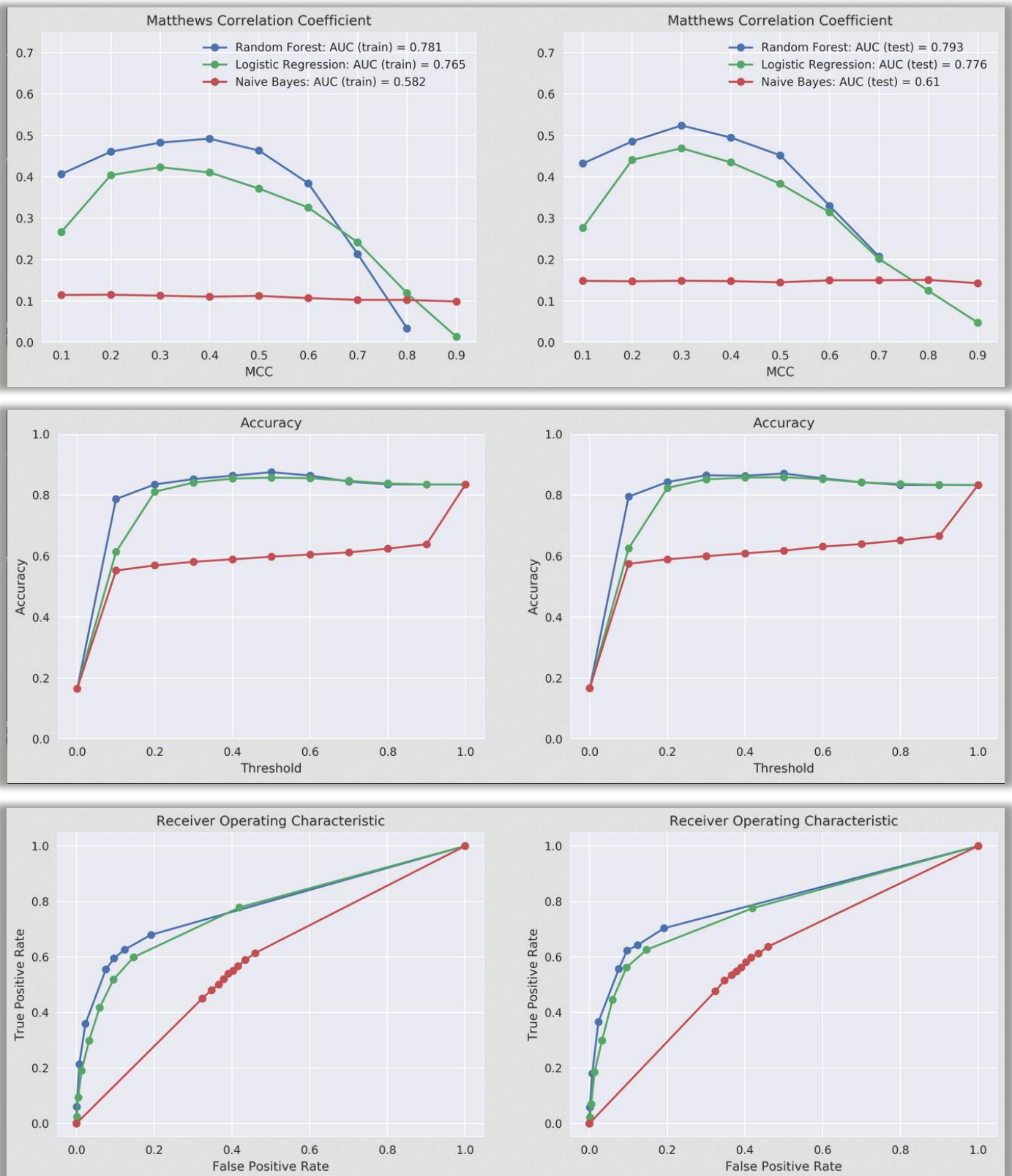
dpi = 300 #dots per square inch when plotting...higher resolution for publications.
plt.rcParams['figure.dpi']= dpi
plt.rcParams["figure.figsize"] = [4,1]

plt.rcParams['figure.figsize'] = (15,5)
plt.subplot(121)
plt.plot(thresholds, mcc1, '-o')
plt.plot(thresholds, mcc2, '-o')
plt.plot(thresholds, mcc3, '-o')
plt.title("Matthews Correlation Coefficient")
plt.xlabel("Threshold")
plt.xlabel("MCC")
plt.ylim(0,0.75)
plt.legend([m1Name + ": AUC (train) = "+ str(round(auc1,3)),
m2Name + ": AUC (train) = "+ str(round(auc2,3)),
m3Name + ": AUC (train) = "+ str(round(auc3,3))])
plt.subplot(122)
plt.plot( thresholds, mcc1Test, '-o')
plt.plot( thresholds, mcc2Test, '-o')
plt.plot( thresholds, mcc3Test, '-o')
plt.title("Matthews Correlation Coefficient")
plt.xlabel("Threshold")
plt.xlabel("MCC")
plt.ylim(0,0.75)
plt.legend([m1Name + ": AUC (test) = "+ str(round(auc1Test,3)),
m2Name + ": AUC (test) = "+ str(round(auc2Test,3)),
m3Name + ": AUC (test) = "+ str(round(auc3Test,3))])
plt.show()
plt.subplot(121)
plt.plot(thresholds, acc1, '-o')
plt.plot(thresholds, acc2, '-o')
plt.plot(thresholds, acc3, '-o')
plt.title("Accuracy")
```

```

plt.ylabel("Accuracy")
plt.xlabel("Threshold")
plt.ylim(0,1.0)
plt.subplot(122)
plt.plot(thresholds, acc1Test, '-o')
plt.plot(thresholds, acc2Test, '-o')
plt.plot(thresholds, acc3Test, '-o')
plt.title("Accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Threshold")
plt.ylim(0,1.0)
plt.show()
plt.subplot(121)
plt.title("Receiver Operating Characteristic")
plt.plot(fpr1, tpr1, '-o')
plt.plot(fpr2, tpr2, '-o')
plt.plot(fpr3, tpr3, '-o')
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.subplot(122)
plt.title("Receiver Operating Characteristic")
plt.plot(fpr1, tpr1Test, '-o')
plt.plot(fpr2, tpr2Test, '-o')
plt.plot(fpr3, tpr3Test, '-o')
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.show()
/root/anaconda2/envs/py35/lib/python3.5/site-packages/matplotlib/font_manager.py:1297:
UserWarning: findfont: Font family ['serif'] not found. Falling back to DejaVu Sans
(prop.get_family(), self.defaultFamily[fonttext]))

```



Based on the plots, which model do you think will best predict heart failure? What is the optimal threshold setting?

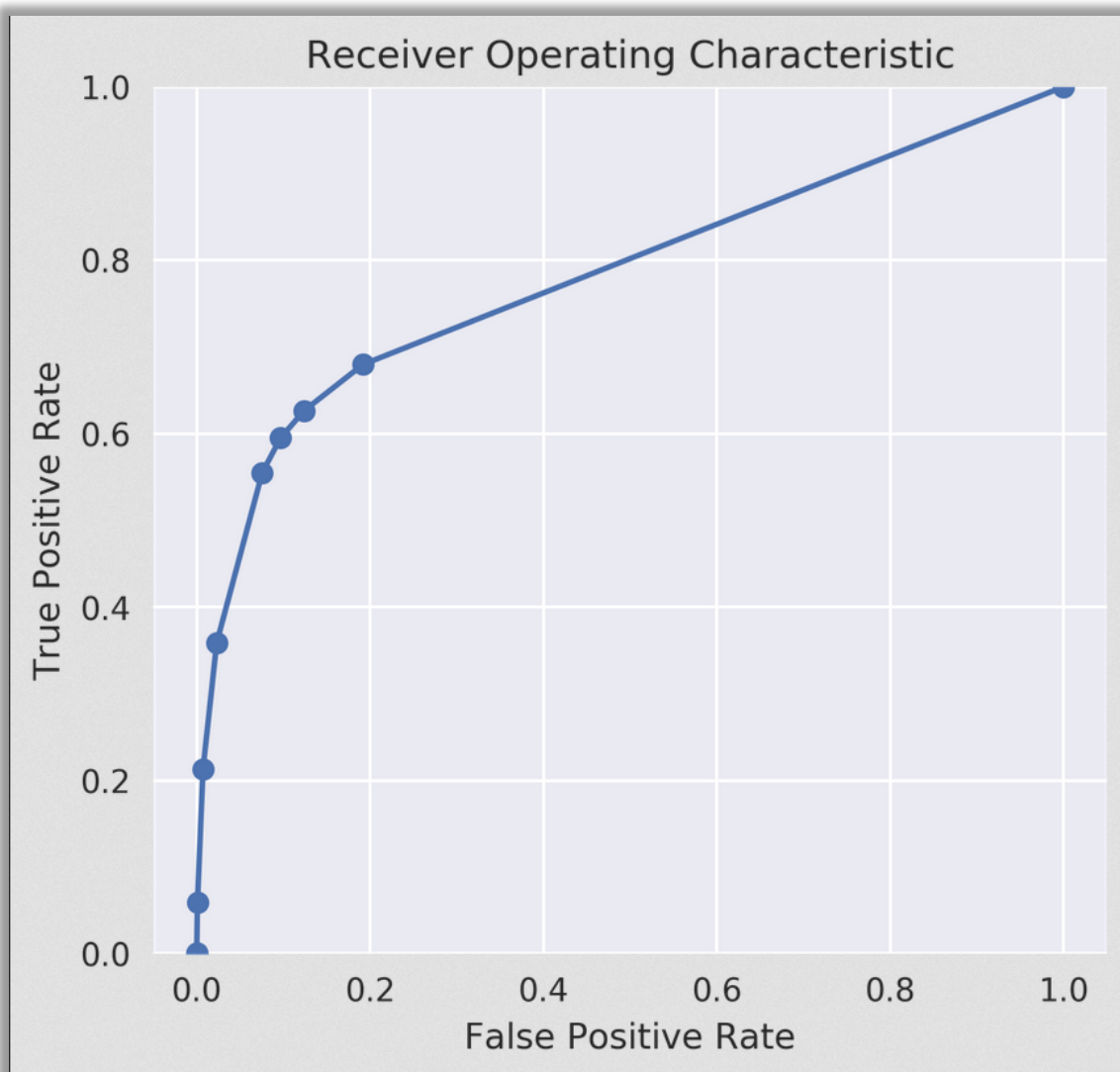
In [25]:

```
plt.title("Receiver Operating Characteristic")
plt.plot(fpr1, tpr1, '-o')
plt.ylabel("True Positive Rate")
plt.xlabel("False Positive Rate")
plt.axis('square')
plt.grid('on')
plt.ylim((0,1))
```

Out[25]:

(0, 1)

```
/root/anaconda2/envs/py35/lib/python3.5/site-packages/matplotlib/font_manager.py:1297:
UserWarning: findfont: Font family ['serif'] not found. Falling back to DejaVu Sans
(prop.get_family(), self.defaultFamily[fonttext]))
```





---

© Copyright IBM Corporation 2019.

The information contained in these materials is provided for informational purposes only and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).



Please Recycle

---