Stroke Detection using Spark MLlib

Brandon Burks, Marco Maldonado, Antonio Santillan, Michael Maldonado

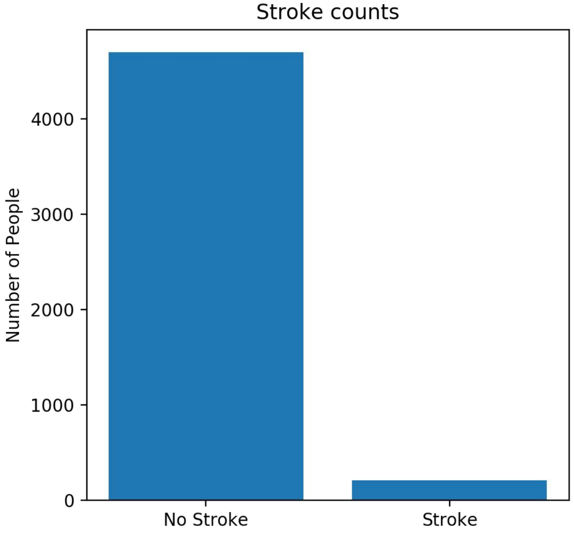
Introduction to Topic

Stroke Prediction

- Important Topic in both Computer Science and Medicine
- Difficult problem due to many factors
- Reduces costs and unnecessary deaths

Dataset Used

- 5110 people, 4909 after cleaning
- 10 features relating to a person's health and livelihood
- 1 if they had a stroke, 0 otherwise
- Overwhelming number of people without stroke



Dataset Before and After Cleaning

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	67.0	0	1	1	0	0	228.69	36.6	2	1
2	0	80.0	0	1	1	0	1	105.92	32.5	0	1
3	1	49.0	0	0	1	0	0	171.23	34.4	3	1
4	1	79.0	1	0	1	1	1	174.12	24.0	0	1
5	0	81.0	0	0	1	0	0	186.21	29.0	2	1

Insights between people with and without strokes

	age	avg_glucose_level	bmi
count	209.000000	209.000000	209.000000
mean	67.712919	134.571388	30.471292
std	12.402848	62.462047	6.329452
min	14.000000	56.110000	16.900000
25%	58.000000	80.430000	26.400000
50%	70.000000	106.580000	29.700000
75%	78.000000	196.920000	33.700000
max	82.000000	271.740000	56.600000

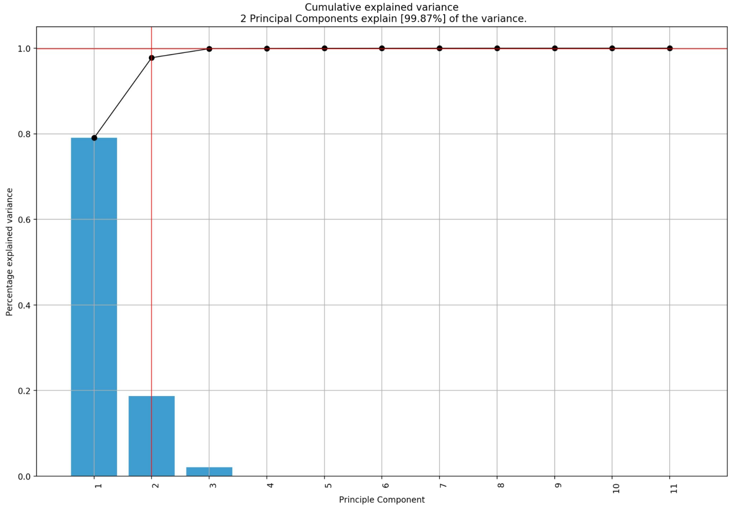
Stroke

	age	avg_glucose_level	bmi
count	4700.000000	4700.000000	4700.000000
mean	41.760451	104.003736	28.823064
std	22.268129	42.997798	7.908287
min	0.080000	55.120000	10.300000
25%	24.000000	76.887500	23.400000
50%	43.000000	91.210000	28.000000
75%	59.000000	112.432500	33.100000
max	82.000000	267.760000	97.600000

No Stroke

Feature importance

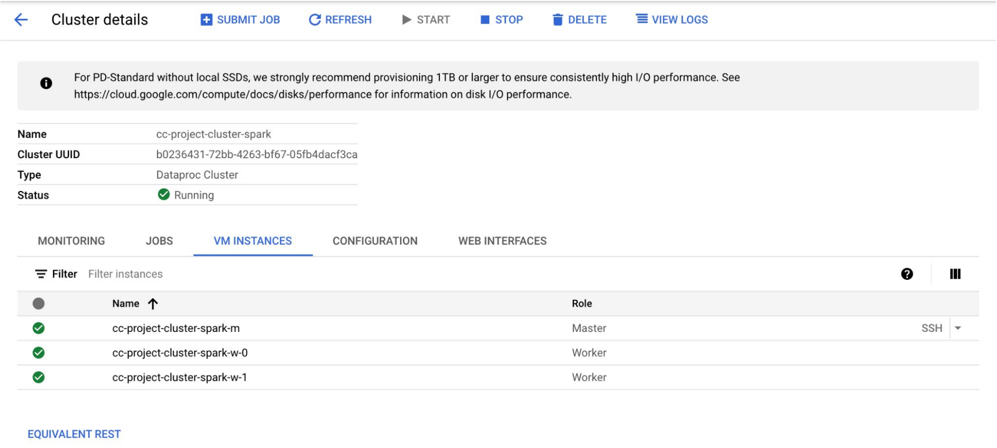
- Using PCA to calculate which features are most important
- Issues due to how we encoded choices (many ones and zeros)
- Top three were: avg_glucose_level, age, and gender



How we utilized Google Cloud Services

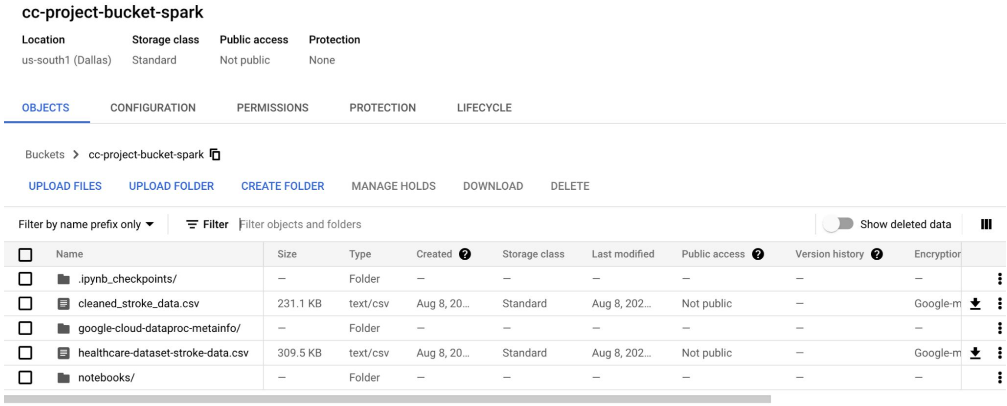
Using Dataproc clusters

- Cluster of smaller VMs that can run mapreduce jobs like Hadoop or Spark
- Can scale based on demand
- Allows use with jupyter notebook for easy data analysis



Setting up Data bucket

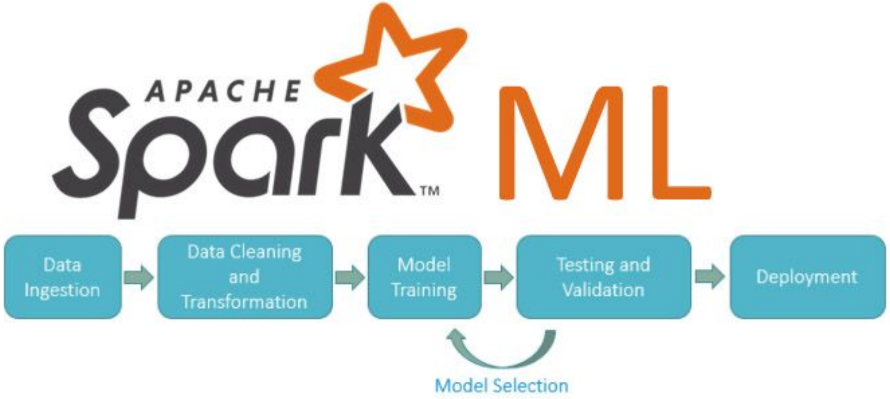
- Stores Notebook file and data to be read into spark
- Allows for access from the Dataproc cluster without authentication



Our implementation in Spark using MLlib

What is MLlib

- Allows machine learning models to scale with large clusters
- Can quickly process large datasets
- Provides various tools to manipulate and analyze data



Logistic Regression

- Classification model for Multinomial/Binary Logistic Regression using Limited-memory BFGS.
- RDD Based Model

```
# Build the model (train the model with parsedDatas)
model = LogisticRegressionWithLBFGS.train(data_spark)
```

Random Forest

- Random Forest learning algorithm for classification
- Dataframe based model

```
data_RF = spark.createDataFrame(data)

assembler = VectorAssembler(inputCols = feature_cols,
                            outputCol = "features")

data_RF = assembler.transform(data_RF)
data_RF = data_RF.select(["features", "stroke"])

train_RF, test_RF = data_RF.randomSplit([0.8, 0.2])
print(f"Training Samples: {train_RF.count()}, Testing Samples: {test_RF.count()}")
```

```
Training Samples: 3955, Testing Samples: 954
```

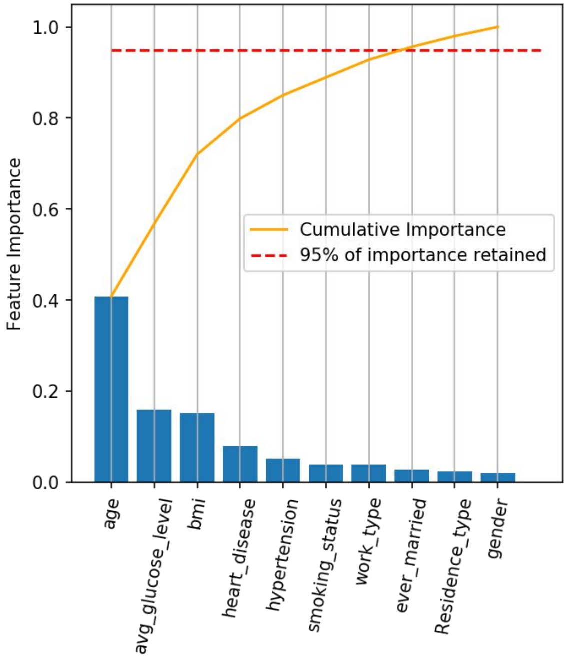
Creating Random forest classifier and training on data

```
RF_classifier = RandomForestClassifier(labelCol = "stroke", numTrees = 60).fit(train_RF)
RF_predictions = RF_classifier.transform(test_RF)
```

Model Results

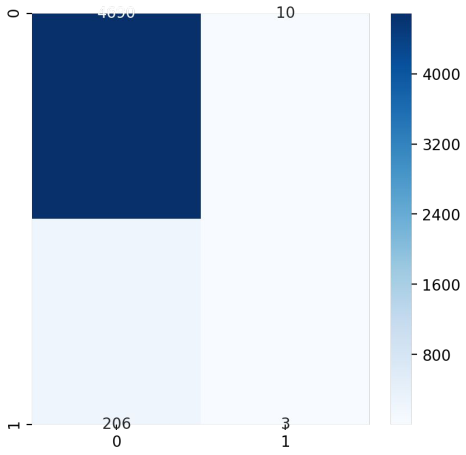
Random Forest results

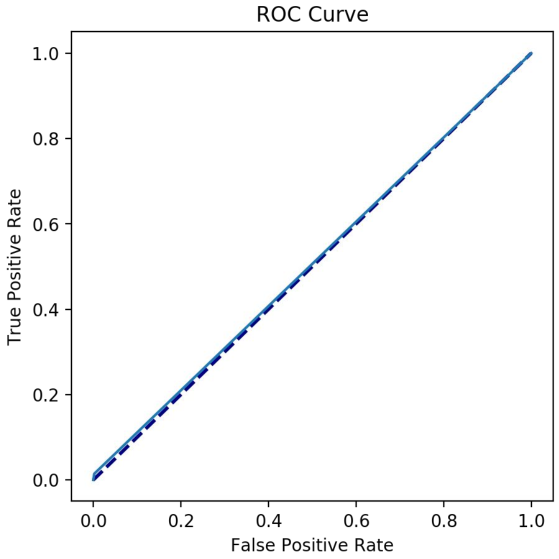
- Testing Accuracy: 0.81
- Difference in Feature importance
    - Top 3: age, avg_glucose_level, bmi



Logistic Regression

- Had a final testing accuracy of 0.955
    - Larger than random forest model
- Limited metrics due to binarization of features





Thank You!