Cloud Computing

Report on subject:
Death from heart failure predictions based on simple machine learning set on cloud

Rafał Kołaczkowski

Student ID number: 292781

Handing in date: 25.01.2022

1. About project

Healthcare has been increasingly changing in past years, two of many contributing factors being interesting from a standpoint of data science engineers – sharing healthcare information across multiple systems and using machine learning algorithms for various applications.

Aim of that project is to build machine learning oriented algorithm used for predicting death from heart failure. That algorithm is based on dataset containing 12 features that can be used to predict mortality by heart failure. In project has been used cloud computing in Azure Databricks.

2. Setting up cluster

To set up cluster used to compute problem, creating Azure Microsoft account has been necessary. Early on, account has had Student status providing money to spend on Azure applications, but because of problems with setting up cluster with the minimum of 8 cores, half for worker type and second half for driver, account has been upgraded with beginning money saved. Next step was to create Azure Databricks and setting up cluster. Worker type and driver type had been set to the least expensive version possible, Standard_F4, which has 8 gigabytes memory and 4 cores, combining 16 gigabytes memory and 8 cores had been allocated in cluster.

3. Dataset

As stated in paragraph 1, used dataset consists of 12 features used as an input and 1 feature, being death event as an output. Dataset contains information about 299 patients.

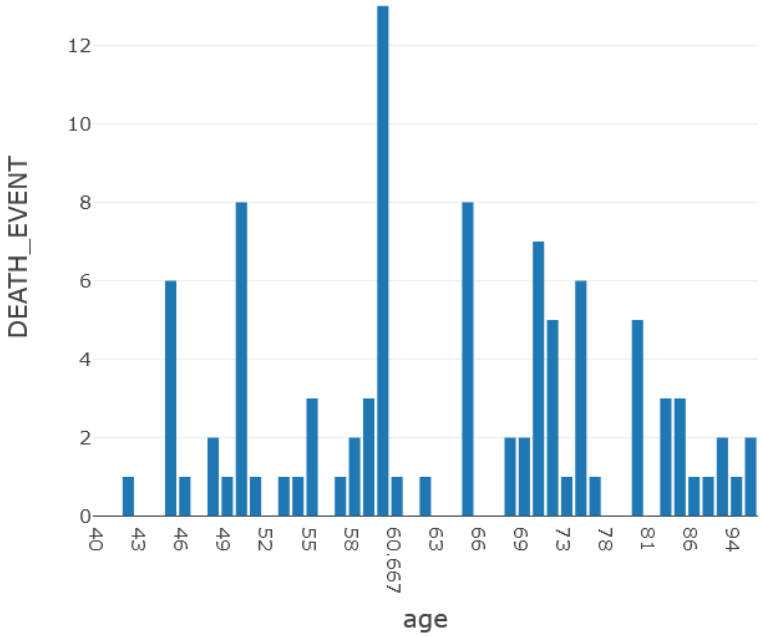
	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
1	582	0	20	1	265000	1.9	130	1	0	4	1
2	7861	0	38	0	263358.03	1.1	136	1	0	6	1
3	146	0	20	0	162000	1.3	129	1	1	7	1
4	111	0	20	0	210000	1.9	137	1	0	7	1
5	160	1	20	0	327000	2.7	116	0	0	8	1
6	47	0	40	1	204000	2.1	132	1	1	8	1
7											

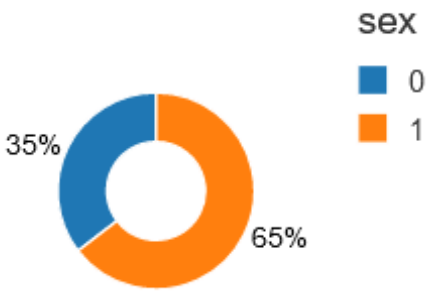
Showing all 299 rows.

Input features are age, anaemia, creatinine phosphokinase, diabetes, ejection fraction, high blood pressure, platelets, serum creatinine, serum sodium, sex, smoking, time.

That dataset has been downloaded from [1] in .csv format. Because of the whole dataset consisting of integer and double values, it is sadly almost impossible to decode how values corresponds to actual health informations, for example it is unknown, whether sex = 1 being male or female. On the other hand dataset consisting only of numbers helps a little bit in setting up machine learning algorithm because of lack of string values.

Using graph libraries included in Azure Databricks, there where created couple graphs showing the data:





4. Databricks notebook

In Azure Databricks many libraries are included, for example DataFrames which provides Spark SQL, GraphX, which provides simple graph creating, MLib which provides machine learning algorithms.

Notebook has been set up to python language, specifically to PySpark.

4.1. Ingesting data to notebook

After uploading the dataset to Azure Databricks, the dataset is being read:

```
1  data = spark.read.csv('/FileStore/tables
   /heart_failure_clinical_records_dataset-1.csv',header=True, sep=',',
   inferSchema=True)
```

4.2. Dataset development

To use input features in machine learning algorithms included in MLib, they have to take form of a vector, so those features are merged and added to a new array which includes all the previous columns. New vector is called ‘features’ and label ‘DEATH_EVENT’ is changed to ‘label’ for more understandable usage.

```
from pyspark.ml.feature import VectorAssembler
train = VectorAssembler(inputCols = ["age", "anaemia",
"creatinine_phosphokinase", "diabetes", "ejection_fraction",
"high_blood_pressure", "platelets", "serum_creatinine", "serum_sodium", "sex",
"smoking", "time", "DEATH_EVENT"], outputCol = "features").transform(data)
train1=train.withColumnRenamed("DEATH_EVENT", "label")
```

New array is split to train data and test data.

```
X,y = train1.randomSplit([0.8, 0.2])
```

4.3. Training models

In that project were used three machine learning algorithms: logistic regression, decision tree and random forest.

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=10, featuresCol = 'features', labelCol =
'label')
lrModel = lr.fit(X)
```

```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol="features", labelCol="label")
dtModel = dt.fit(X)
```

```
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(labelCol="label", featuresCol="features",
numTrees=1)
rfModel = rf.fit(X)
```

4.4. Test models and accuracy

After using 80% of the data to train, the rest was used to test accuracy of the models.

```
lr_prediction = lrModel.transform(y)
dt_prediction = dtModel.transform(y)
rf_prediction = rfModel.transform(y)
```

```
from sklearn.metrics import confusion_matrix
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
```

```
lr_acc = evaluator.evaluate(lr_prediction)
print("Logistic Regression prediction Accuracy: ", lr_acc)

y_pred=lr_prediction.select("prediction").collect()
y_orig=lr_prediction.select("label").collect()

cm = confusion_matrix(y_orig, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
dt_acc = evaluator.evaluate(dt_prediction)
print("Decision Tree prediction Accuracy: ", dt_acc)

y_pred=dt_prediction.select("prediction").collect()
y_orig=dt_prediction.select("label").collect()

cm = confusion_matrix(y_orig, y_pred)
print("Confusion Matrix:")
print(cm)
```

```
rf_acc = evaluator.evaluate(rf_prediction)
print("Random Forest prediction Accuracy: ", rf_acc)

y_pred=rf_prediction.select("prediction").collect()
y_orig=rf_prediction.select("label").collect()

cm = confusion_matrix(y_orig, y_pred)
print("Confusion Matrix:")
print(cm)
```

Accuracy of model are counted as test area under ROC.

```
Logistic Regression prediction Accuracy:  1.0
Confusion Matrix:
[[42  0]
 [ 0 17]]
Decision Tree prediction Accuracy:  1.0
Confusion Matrix:
[[42  0]
 [ 0 17]]
Random Forest prediction Accuracy:  1.0
Confusion Matrix:
[[42  0]
 [ 0 17]]
```

As can be seen, accuracy of every model is 100% which seems to be overly accurate, so it is expected that mistakes have been made during the project, unfortunately they have not been spotted. Reducing maximum numbers of iteration in logistic regression makes accuracy drop to 0.998, which is still overly accurate.

5. Conclusion

Because of new ways to approach big datasets in form of easily accessible files in cloud, machine learning algorithms, parallel computing provided in platforms like Azure Databricks or AWS, many different possibilities are open, from much more client adjusted advertisements to, even as shown in that project, healthcare datasets management and predictions based on them, which would help in patients diagnosing.

6. References

[1] https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

[2] https://spark.apache.org/docs/latest/api/python/

During the project work, has been used many different tutorials, articles and datasets, which were not included in references, because their contribution wasn't big even though they provided some help.