Production Line Performance

Project work in Languages and Algorithms for Artificial Intelligence class (Module 2)

Alessio Falai
alessio.falai@studio.unibo.it

July 31, 2020

Alma Mater Studiorum - University of Bologna

Table of contents

page

Datasets

- **Arrest**: Contains statistics, in arrests per 100.000 residents, for assault, murder, and rape in each of the 50 US states in 1973. It also gives the percent of the population living in urban areas.
- **Adult**: Aims at separating people whose income is greater than 50 thousands dollars per year from the rest.

- **Arrest**: Contains statistics, in arrests per 100.000 residents, for assault, murder, and rape in each of the 50 US states in 1973. It also gives the percent of the population living in urban areas.
- **Adult**: Aims at separating people whose income is greater than 50 thousands dollars per year from the rest.

- **Bosch**: Aims at predicting internal failures using thousands of measurements and tests made for each component along different assembly lines.

Preprocessing

Data exploration

- The Bosch dataset includes 3 subsets: numerical, categorical and time data.
- As stated in [2], the categorical data is extremely sparse, and thus not exploited in subsequent stages.
- Our first analysis will be focused on just numerical data:
  - 968 anonymized features
  - 1 183 747 labeled examples
  - 0.58% of failed products
  - 78.5% of missing values

- The Bosch dataset includes 3 subsets: numerical, categorical and time data.
- As stated in [2], the categorical data is extremely sparse, and thus not exploited in subsequent stages.
- Our first analysis will be focused on just numerical data:
  - 968 anonymized features
  - 1 183 747 labeled examples
  - 0.58% of failed products
  - 78.5% of missing values

- The Bosch dataset includes 3 subsets: numerical, categorical and time data.
- As stated in [2], the categorical data is extremely sparse, and thus not exploited in subsequent stages.
- Our first analysis will be focused on just numerical data:
  - 968 anonymized features
  - 1 183 747 labeled examples
  - 0.58% of failed products
  - 78.5% of missing values

Data exploration

- The Bosch dataset includes 3 subsets: numerical, categorical and time data.
- As stated in [2], the categorical data is extremely sparse, and thus not exploited in subsequent stages.
- Our first analysis will be focused on just numerical data:
  - 968 anonymized features
  - 1 183 747 labeled examples
  - 0.58% of failed products
  - 78.5% of missing values

- **Stage I**: This step clusters data with similar processes together into process groups.
- **Stage II**: This step uses supervised learning to predict the failed products. Each cluster is treated as an independent dataset and has its own classifier.

- **Stage I**: This step clusters data with similar processes together into process groups.
- **Stage II**: This step uses supervised learning to predict the failed products. Each cluster is treated as an independent dataset and has its own classifier.

Data preprocessing

- **Common**: Columns containing null values and constant values in each row are dropped.
- **Clustering**: Values get binarized (0 meaning null value and 1 meaning not null value) and PCA is applied to binarized features.
- **Classification**: A feature imputation method (mean value over the column), followed by feature standardization (zero mean, unit variance) and PCA, is applied over non-binary values in each cluster.
- **Prediction**: A new example follows the same preprocessing scheme as the whole dataset.

Data preprocessing

- **Common**: Columns containing null values and constant values in each row are dropped.
- **Clustering**: Values get binarized (0 meaning null value and 1 meaning not null value) and PCA is applied to binarized features.
- **Classification**: A feature imputation method (mean value over the column), followed by feature standardization (zero mean, unit variance) and PCA, is applied over non-binary values in each cluster.
- **Prediction**: A new example follows the same preprocessing scheme as the whole dataset.

- **Common**: Columns containing null values and constant values in each row are dropped.
- **Clustering**: Values get binarized (0 meaning null value and 1 meaning not null value) and PCA is applied to binarized features.
- **Classification**: A feature imputation method (mean value over the column), followed by feature standardization (zero mean, unit variance) and PCA, is applied over non-binary values in each cluster.
- **Prediction**: A new example follows the same preprocessing scheme as the whole dataset.

Data preprocessing

- **Common**: Columns containing null values and constant values in each row are dropped.
- **Clustering**: Values get binarized (0 meaning null value and 1 meaning not null value) and PCA is applied to binarized features.
- **Classification**: A feature imputation method (mean value over the column), followed by feature standardization (zero mean, unit variance) and PCA, is applied over non-binary values in each cluster.
- **Prediction**: A new example follows the same preprocessing scheme as the whole dataset.

Clustering

The chosen clustering algorithm is k-means, since density-based methods (like DBSCAN) are still not available in Spark. So, the following issues needed to be addressed:

- **Problem**: Automatically select the right amount of clusters:
  - Silhouette and elbow methods are not ideal, since they require human analysis.
  - Spark 3.0.0 dropped support for inertia computation, maintaining only evaluation by silhouette scores.
- **Solution**: Ad-hoc implementation of the Gap statistic method, described in [1].

The chosen clustering algorithm is k-means, since density-based methods (like DBSCAN) are still not available in Spark. So, the following issues needed to be addressed:

- **Problem**: Automatically select the right amount of clusters:
  - Silhouette and elbow methods are not ideal, since they require human analysis.
  - Spark 3.0.0 dropped support for inertia computation, maintaining only evaluation by silhouette scores.
- **Solution**: Ad-hoc implementation of the Gap statistic method, described in [1].

The chosen clustering algorithm is k-means, since density-based methods (like DBSCAN) are still not available in Spark. So, the following issues needed to be addressed:

- **Problem**: Automatically select the right amount of clusters:
  - Silhouette and elbow methods are not ideal, since they require human analysis.
  - Spark 3.0.0 dropped support for inertia computation, maintaining only evaluation by silhouette scores.
- **Solution**: Ad-hoc implementation of the Gap statistic method, described in [1].

Gap statistic

- For $k = 1, .., K$, where K is the maximum number of clusters, perform k-means and compute the resulting inertia I_k.
- Generate B reference datasets by sampling from a uniform distribution over each feature, where the support is directly identified by features ranges. Then, for $k = 1, .., K$ and $b = 1, .., B$, perform k-means and compute the resulting inertia I_{kb}. Finally, estimate $E^*[log(I_{kb})]$ as $\frac{1}{B}\sum_b log(I_{kb})$.
- Compute the Gap score as $Gap(k) = E^*[log(I_{kb})] - log(I_k)$.
- Compute the standard deviation sd_k of $log(I_{kb})$ and define $s_k = sd_k * \sqrt{1 + \frac{1}{B}}$.
- Select the minimum k s.t. $Gap(k) - Gap(k+1) + s_{k+1} \geq 0$.

Gap statistic

- For $k = 1, .., K$, where K is the maximum number of clusters, perform k-means and compute the resulting inertia I_k.
- Generate B reference datasets by sampling from a uniform distribution over each feature, where the support is directly identified by features ranges. Then, for $k = 1, .., K$ and $b = 1, .., B$, perform k-means and compute the resulting inertia I_{kb}. Finally, estimate $E^*[log(I_{kb})]$ as $\frac{1}{B}\sum_b log(I_{kb})$.
- Compute the Gap score as $Gap(k) = E^*[log(I_{kb})] - log(I_k)$.
- Compute the standard deviation sd_k of $log(I_{kb})$ and define $s_k = sd_k * \sqrt{1 + \frac{1}{B}}$.
- Select the minimum k s.t. $Gap(k) - Gap(k+1) + s_{k+1} \geq 0$.

Gap statistic

- For $k = 1, .., K$, where K is the maximum number of clusters, perform k-means and compute the resulting inertia I_k.
- Generate B reference datasets by sampling from a uniform distribution over each feature, where the support is directly identified by features ranges. Then, for $k = 1, .., K$ and $b = 1, .., B$, perform k-means and compute the resulting inertia I_{kb}. Finally, estimate $E^*[log(I_{kb})]$ as $\frac{1}{B}\sum_b log(I_{kb})$.
- Compute the Gap score as $Gap(k) = E^*[log(I_{kb})] - log(I_k)$.
- Compute the standard deviation sd_k of $log(I_{kb})$ and define $s_k = sd_k * \sqrt{1 + \frac{1}{B}}$.
- Select the minimum k s.t. $Gap(k) - Gap(k+1) + s_{k+1} \geq 0$.

Gap statistic

- For $k = 1, .., K$, where K is the maximum number of clusters, perform k-means and compute the resulting inertia I_k.
- Generate B reference datasets by sampling from a uniform distribution over each feature, where the support is directly identified by features ranges. Then, for $k = 1, .., K$ and $b = 1, .., B$, perform k-means and compute the resulting inertia I_{kb}. Finally, estimate $E^*[log(I_{kb})]$ as $\frac{1}{B}\sum_b log(I_{kb})$.
- Compute the Gap score as $Gap(k) = E^*[log(I_{kb})] - log(I_k)$.
- Compute the standard deviation sd_k of $log(I_{kb})$ and define $s_k = sd_k * \sqrt{1 + \frac{1}{B}}$.
- Select the minimum k s.t. $Gap(k) - Gap(k+1) + s_{k+1} \geq 0$.

Gap statistic

- For $k = 1, .., K$, where K is the maximum number of clusters, perform k-means and compute the resulting inertia I_k.
- Generate B reference datasets by sampling from a uniform distribution over each feature, where the support is directly identified by features ranges. Then, for $k = 1, .., K$ and $b = 1, .., B$, perform k-means and compute the resulting inertia I_{kb}. Finally, estimate $E^*[log(I_{kb})]$ as $\frac{1}{B}\sum_b log(I_{kb})$.
- Compute the Gap score as $Gap(k) = E^*[log(I_{kb})] - log(I_k)$.
- Compute the standard deviation sd_k of $log(I_{kb})$ and define $s_k = sd_k * \sqrt{1 + \frac{1}{B}}$.
- Select the minimum k s.t. $Gap(k) - Gap(k+1) + s_{k+1} \geq 0$.

Classification

Classifiers

- Decision Tree, **Random Forest**, Gradient Boosted Tree models.
- Hyper-parameters selected by cross-validation on a parameter grid.

Different evaluation strategies, based on a custom confusion matrix computation:

- **Accuracy**: $\frac{TP+TN}{TP+FP+FN+TN}$ (not well-suited to the Bosch dataset, given its high class imbalance).
- **F_1-score**: $2 \times \frac{prec \times rec}{prec+rec}$, where $prec = \frac{TP}{TP+FP}$ and $rec = \frac{TP}{TP+FN}$.
- **Matthew's Correlation Coefficient (MCC)**: $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}}$.
- **Area under ROC**: The two-dimensional area underneath the entire ROC curve ($sens/rec$ at varying thresholds) from $(0,0)$ to $(1,1)$, where $sens = \frac{FP}{FP+TN}$.

Execution

Cloud

- Spark 3.0.0 is not yet available on AWS EMR service (the last version that can be used is 2.4.4).
- Spark 3.0.0 could be manually installed to EC2 clusters, but the Flintrock service already provides some shortcuts to create the desidered clusters and automatically install the selected Spark/Hadoop version.
- Unfortunately, Flintrock was not tested with Spark 3.0.0, which in some installations relies on Java 11 (as opposed to Spark 2.4.4 which simply uses Java 8).
- **Cloud machine**:
  - **Type**: `t2.xlarge`
  - **CPU**: 4 vCPUs based on Intel Xeon with Intel AVX, Intel Turbo
  - **RAM**: 16 GB

Local

Local machine:

- Type: Macbook Pro 16-inch 2019
- CPU: 2.3 Ghz 8-Core Intel Core i9
- RAM: 16 GB 2667 MHz DDR4

Thank you for your attention

References

Robert Tibshirani, Guenther Walther, and Trevor Hastie.
Estimating the number of clusters in a dataset via the gap statistic.
63:411–423, 2000.

Darui Zhang, Bin Xu, and Jasmine Wood.
Predict failures in production lines: A two-stage approach with clustering and supervised learning.
pages 2070–2074, 12 2016.

Backup frame

This is a backup frame, useful to include additional material for questions from the audience.