Polytechnic University of Catalonia

Algorithms, Data Structures and Databases

**Data Science End-to-End Project**

**Part II. The Data Analysis Backbone**

| ***Authors***  Chriki, Fatima Zohra  Xu, Ange | ***Teachers***  Anna.Queralt  Oscar,Romero |
| --- | --- |



January ◇ QT - 2022/23

Instructions of how to access different platforms

In the following links we can find the project repository for the both execution environments.

**Development platform:**

<https://drive.google.com/drive/folders/1ofpTnAzdecSDCze-JZK5ZNnAOZYtokDs?usp=share_link>

**Operation platform:**

<https://github.com/AngeXu8/Startup-success-and-school-performance>

Index

[**1. CONTEXT**](#_svhuc6b5x74d) **4**

[**2. THE DATA ANALYSIS BACKBONE**](#_21or8ogsiaq7) **4**

[**3. ADVANCED TOPIC**](#_jg6khoc2okw2) **7**

[**4. OPERATIONS**](#_l26ocxy0nbx) **8**

[**5. PROJECT ARCHITECTURE EVALUATION**](#_t8ckh344iyiy) **10**

[**6. CONSTRAINTS FULFILLMENT**](#_ljytfhi4m8lx) **12**

[**7. ANALYSIS RESULTS**](#_onzjqttgf9i9) **13**

# CONTEXT

On the one hand, every year many start-ups begin their business with innovative ideas but only a small proportion of these survive in the long run. On the other hand, according to researchers, the school quality and student performance can vary widely within cities. In cities where students of certain profiles concentrated (in foster care, immigrants, with economic problems, etc.) in some schools could result in a higher variation in school quality.

The aim of this project is to analyze if the school quality within cites could be one of the factors that lead to startup success and forecast a company’s success in the United States.

**Original data sources**

For this project we extracted from different sources 4 datasets that can be used for our analysis.

* The School State Assessments 2018
* US Public Schools Dataset
* US Private Schools Dataset
* StartUp Investments

All these four datasets were treated, preprocessed and merged in the Data Management Backbone and, as a result, we generated one final dataset called “startups\_students Performance”. This dataset includes data about different startups and college student’s performances in different cities located in the US.

**Analysis questions**

For this second part of the project, as stated before, our goal is to respond the following analysis questions:

* Which are the factors that most influence when predicting the status of a startup?
* If the school quality of a city could be one of the factors that influences the success of the startups in the same city?

# THE DATA ANALYSIS BACKBONE

When developing this part of the project, we skipped the generation of the Analytical Sandbox zone as for our analytical questions there is no need to select a subset of features or columns from the exploitation zone, because the final dataset created in the previous part already contains enough data to perform our analysis. Thus, our analysis backbone is organized as follows:

**DATABASES**

**Feature Generation Zone**

* **DB\_FeatureGeneration:** includes all the data generated in this zone. All the notebooks of this zone read and write in this dataset.

**Model Training Zone & Testing Zone**

* **DB\_Train&Test:** includes the training data and testing data generated from the Feature Generation Zone. All the notebooks of Model Training and Testing Zone use this database to read the train/testing data.

**NOTEBOOKS**

The notebooks are organized by zones. Each zone represents a folder, and it includes it’s corresponding notebooks:

**Feature Generation Zone**

* **LoadingData:** this script is used to export the dataset stored in the exploitation zone database to the feature generation zone database. The reason for having this is for development purposes, as it can allow us to quickly recover the original dataset from which we began our development, in the case that we accidentally removed or overwritten some features of the actual working dataset.
* **DataPreparation:** the aim of this script is to prepare the data ready to be used for the training. The decision of which features to include in the final dataset that will be split into training and testing datasets. The rule we followed to do the selection was a simple map of the variables to features. We also performed tasks such as data normalization, data cleaning, data imputation, etc.
* **FeatureTransformation:** the goal is to improve the data features quality, as a bad quality of features can produce a negative effect on the model performance. Firstly, we applied transformations to several features into new features by using functions like log to better fit for normal distribution, as one of the important assumptions when building models for analysis is the normality. By doing so, we improve the data quality and obtain better results in the model training. Then we generated some new features from the existing ones with the belief that the new ones could explain more the target feature, by using multiplication or addition (add interaction between multiple features). For example, from the original features we generated a new feature called nFoundedDays, which is the number of the days passed since the startups was founded when extracting the dataset.
* **FeatureSelection:** notebook that performs feature selection as advanced topic. Details can be found in section 4 of advanced topic.
* **DataSplit:** we split the final cleaned dataset into training (70%) and testing (30%) dataset. We also made sure that the proportion of our target’s classes is balanced in both training and testing dataset, in order to avoid possible bias during the model training.

**Model Training Zone**

* **LoadingData:** this notebook loads the training and testing dataset from the Feature Generation Zone and imports these datasets into a new database for the actual Model Training Zone called DB\_Train&Test.
* **TrainingModel:** as one of the analysis questions of this project is to understand the relationship between features that come from different sources(startups and students performance), among all the machine learning algorithms we chose to use Decision Tree, as it’s easy to interpret the generated tree structure, especially for our case where we want to see which are the important features when predicting the status of a startups taking into account the students performance. Thus, this notebook firstly performs a hyperparameter tuning in order to find out which is the best hyperparameters configuration for our model, depending on the dataset characteristic.

**Model Validation Zone**

* **ModelValidation**: the final model will be validated using the testing dataset in the Model Validation Zone, where techniques such as k-fold cross validation were applied in order to evaluate the generalization capacity of the model on unseen data, and to generate less biased results. We applied several metrics to summarize and take the average performance of the k-fold cross-validation models, such as accuracy, f1-score, recall and precision.
* **ModelVisualization**: this notebook is in charge of showing the model’s visualization results, such as the decision tree plot, the confusion matrix, the ROC-Curve, etc. We have also generated a pdf file, called “ModelVisualization.pdf”, with all the graphics and model performance plots available in this notebook.

**Tools used for each part of the architecture**

In order to carry out this project, we made use of several tools [*Figure 1*]. For development, we used the programming language Python, since it provides all the necessary libraries for the implementation of each zone:

* **Pandas & Numpy:** these libraries provide functions for the generation, transformation and manipulations of data structures. They were mainly used in the Feature Generation Zone.
* **Scikit-Learn:** this library provides several tools to generate and evaluate Machine Learning models. So, it was widely used in the Model Training and Validation Zone.
* **Tkinter**: we used this tool to build a graphic interface for the model deployment.
* **DuckDB**: the communication between zones was possible by using this SQL Data Management System, where we created one database per each zone (except the Model Training and Validation Zones) and all the data transferring process between zones consists only in reading the data from one zone to another zone.
* **Pickle:** provides functions to export and import ML models generated in the Model Training and Validation Zones.

The main IDE (Integrated Development Environment) used for this project was Google Colab, it allows multiple users to share and edit notebooks in python. Also, all the notebooks and datasets generated during the development were stored and shared through Drive. For this reason, Colab was the most convenient tool to use since it allows direct access to the files stored in Drive. In addition to Colab, we also used Jupyter notebooks on several occasions to do local testing.

One limitation we had to face was finding a perfect real-time editor for notebooks. We did a lot of research and the only tool found was DeepNote, which allows real-time and parallel notebook editing. However, in DeepNote it is hard to manage and organize different generated notebooks for each zone of the Data Analysis backbone, thus, we discarded this tool and ended up using Google Colab.

Another limitation we faced was the limited use of resources per user in Colab or when executing some specific python commands since the working environment is based in the cloud, and this is the reason for the use of Jupyter to do local testing.

|  |
| --- |
| **Figure 1:** *Tools used in the Analysis Backbone Project* |

# ADVANCED TOPIC

Our dataset (the dataset used for analysis and stored in the exploitation zone), includes almost 90 features. This is a problem that need to be solved since it has several consequences:

* The higher is the dimension of training data, the higher is the training time and resources required (CPU, RAM, etc). Also, the model may crash due to the high data dimensionality.
* There are irrelevant features which add noise to the training data. This has a negative impact on the model learning and predictive power.
* There is a high risk of overfitting when the model is trained with a high number of features.

The solution to this problem is to apply feature selection. Our dataset includes several irrelevant and redundant features that need to be removed in order to improve the model performance and reduce the training time. There are several techniques to reduce data dimensionality and, after doing a lot of research, we decided to apply Principal Component Analysis (PCA) to our data for the following reasons:

* It is one of the most efficient methods to reduce dimensionality for numeric data. Since almost 80% of our dataset’s features are numeric, there is no problem to apply it.
* PCA reduces the data dimensionality by returning the X principal component generated by merging the input numeric features. It automatically removes redundant features and returns independent components, which are uncorrelated.
* PCA removes the irrelevant and noise features, and with that we reduce overfitting and improve the performance of the model.
* PCA allows visualizing components returned, thus, we can visually see the importance of each feature, the relation between them and identify influential patterns.

Our dataset includes 77 numeric variables, after performing PCA, these features were reduced to 15 principal components. These components explain over 70% of variability of the 77 numeric features, which is good enough for the later training process.

All the feature selection was implemented in one single notebook, located in the Feature Generation Zone (FGZ). We decided to put it in this zone because feature selection needs to be performed before splitting the data into training and testing datasets. This notebook receives a dataset as input, applies PCA and returns a final dataset with lower dimensionality. The notebook was designed to be added (or removed) to the FGZ without conflicts, that is, no adjustments nor configurations are needed to add the new code of this advanced topic to the FGZ. In addition, the feature selection was designed to be applied over any other dataset, that is, it receives a dataset, performs PCA, and automatically returns the reduced dataset. As we stated before, PCA works with numeric data, so the only requirement to execute the script is that the input dataset includes numeric data. Moreover, the rest of the notebooks of FGZ zone are also designed to be executed without adjustment if we add or remove the feature selection notebook. Obviously, the only adjustment needed is the order of execution of the notebooks, since it needs to be executed before splitting the data into training and testing. The order of execution of the FGZ scripts is as follows:

*LoadingData → DataPreparation → DataTransformation → FeatureSelection → DataSplit*

# OPERATIONS

**4.1.** **Environment architecture**

We have organized the operations environment for the data analysis backbone by creating the following folders:

* **Data Management**: in this folder we have all the python scripts converted from notebooks corresponding to the data management backbone in the development environment. We also did modifications and removal on the source code in order to adapt them better for this environment.
* **Data Analysis**: same as the data management folder, here we have all the python scripts corresponding to the data analysis backbone in the development.
* **Development**: in this folder we have the following files:
* **requirements.txt:** it contains all the dependencies needed to execute the script orchestrateAll.py.
* **orchestrateAll.py:** code implemented to orchestrate the execution of all the scripts mentioned in the previous sections.
* **tests.py**: code that performs tests to ensure the correct execution of the pipeline.
* **modelInterface.exe:** the executable of the trained and validated model that will be deployed into production.

**4.2. Model deployment**

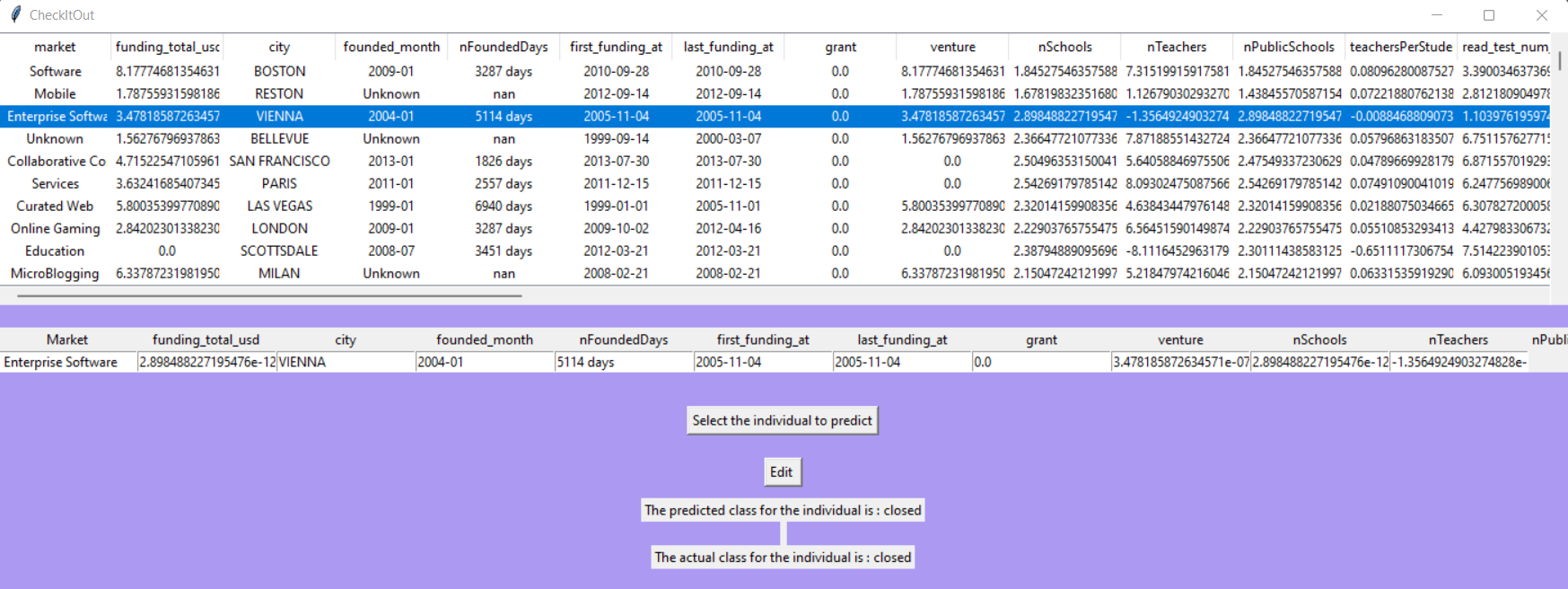
Once we have the model trained and validated, we proceed to put it into production to make it available for decision-making and predictions. For this purpose we implemented a code(orchestrateAll.py) to orchestrate the execution of all the independent code, in this case, they are all the notebooks converted in .py belonging to the data management and analysis backbones. After that, this orchestrated code will be executed and the final built model will be put into the operations environment (Github).

The above workflow mentioned was implemented was based on the CI/CD pipeline, it consists of the following stages:

* **Source**: The source code(orchestrated code and the scripts.py) is pushed to the version control repository(in real life this code change will trigger the CI/CD system and the CI/CD pipeline will be executed).
* **Build**: The source code is executed with its dependencies (we created a requiement.txt that includes all the dependencies needed to be installed for the execution) and it will generate a model executable, that consists of a graphic user interface where the model is available to do predictions and decision makings [*Figure 2*]. This GUI is built with the goal to simulate a real operations environment, in order to do so, we prepared some testing data to avoid having to provide them manually.

The steps to follow to make predictions on this interface are: firstly, we choose from the list of individuals the data point that we want to test, once we have it selected, we click on the button “Select the individual to predict”. After that, the results of the prediction will be printed on the user interface, as depicted in Figure 2. It will tell us which is the predicted status (opening, acquired or closed) of the selected individual (startup) and its real status. This GUI can also allow us to edit the features value of the selected individual by pressing the button “Edit”, by doing so, we are evaluating the prediction capability of the model on unseen data.

In real life, the buildment of the executable is usually done with Docker containers.



**Figure 2:** *Model deployment simulation*

* **Test**: We run tests over the source code to make sure that there are no bugs present in the implementation.
* **Deploy**: The model is ready to be deployed into operations, in this case, as we supposed that our operation environment is Github, the executable of the model will be pushed on the repository.

**Adding the advanced topic**

When there are new changes/functionalities to be included in the pipeline, in this case, is the script implemented for the advanced topic, the only change to the final orchestrate code we had to make was the inclusion of the command that run the advanced topic script, this is thanks to the way we organized all the notebooks of each zone, as during the development tried to make them work as an “independent piece of code” as possible. It results that making modifications or changes to the notebooks usually does not imply making adjustments in the rest of the notebooks.

Executing the whole notebooks of Feature Generation Zone, following the mentioned order, we get:

* [Without Feature Selection]: A cleaned training and testing datasets ready to be used in the Model Training and Validation Zone.
* [With Feature Selection]: A cleaned training and testing datasets with lower dimension ready to be used in the Model Training and Validation Zone.

# PROJECT ARCHITECTURE EVALUATION

**DYNAMICITY**

***How easy is it to add a new variable into an existing source? And add a new source?***

* As for the addition of a new variable into the existing source, we have different scenarios:
* For the data management backbone: we will need to modify the notebook that is in charge to do the merging of different versions of a dataset into one in the Trusted Zone.
* For data analysis backbone: if we consider that it is important and necessary to include the new variable for the model training and analysis, we will need to modify or add the notebook in the Analytical SandBox that selects the columns or features from the Exploitation Zone.

In general, the addition of a new variable implies to change or add one notebook.

* As for the addition of a data source, we have different scenarios:
* For the data management backbone: we will need to add notebooks that perform some specific tasks for the new dataset, such as data quality (Trusted Zone) and the merging of this dataset with the existing ones (Exploitation Zone). In general, this addition does imply the modification of other notebooks.
* For data analysis backbone: same as before, if we consider that the new dataset contains relevant information for our analysis, we will need to modify or add the notebook in the Analytical SandBox that selects the columns or features from the Exploitation Zone.

***How easy is it to change the transformations executed between two zones?***

In our design, each zone has its own database. Thus, if we are doing changes in a specific zone, these changes will be reflected only in the database of the corresponding zone. If we want to propagate the changes (transfer the data to the next zone), we can use the *LoadingData* script to load the data from the previous zone to the next zone. The goal of this design is to guarantee the separability of data of each zone.

**REUSABILITY**

For some common data treatment that can be applied for any dataset, we created a general notebook for common tasks and if needed, notebooks for specific goals will be created. For example, in the Trusted Zone we created general notebooks such as data profiling, data cleaning, etc.

**OPENNESS**

In the previous section of the advanced topic we explained how easy it is to extend and evolve the actual architecture with new / advanced aspects.

**REPRODUCIBILITY**

We created a script (orchestrateAll.py) that automates the execution of all the scripts of each zone (from landing to deployment). So, as input we have raw data (datasets in landing), and as a result we have an executable (deployment), which consists of an interface that allows us to make predictions of the status of startups using the generated ML model.

Moreover, we generated a requirements.txt file that includes all the dependencies needed to run the orchestrate script. We also added a README file to guide us.

**RIGOROUS THINKING**

* As for the operations layer, we noticed that with the actual project architecture, once we deployed the model, it began to work with real life data, which is constantly changing due to many reasons that make the actual model not suitable for the business need. In this situation, we need a monitoring tool in the operations environment that periodically will evaluate and record the performance of the model and take actions when it is performing poorly.
* For this second part of the project we only had to create one data analysis backbone for one analytical purpose. However, in real life, there will be more than one analysis backbones and the Exploitation Zone will have to provide data sources for all of them. This could produce a problem of scalability and storage if we have the data stored in on-premise servers. On one hand, in the worst case, if all the backbones read data sources from this zone, this could lead to the problem of having a low bandwidth, especially for users in geolocations that are far away from the location of the servers, the bandwidth of reading the data sources would be affected negatively. On the other hand, the maintenance and storage of a large amount of data is also cost-consuming, especially when data outages happen. To solve these problems, we thought that cloud-based services are the best solution, such as AWS or Azure, as they are cost saving and guarantee data loss prevention, advanced security and other advantages.
* During the project development, we had some incompatibilities problems with some python libraries versions. So, the libraries used in the projects may be updated again and that may cause error executions. So we think there must be a monitoring control in order to avoid those errors.
* We noticed that, when we want to run the code, we have to transfer it from one environment to another, and that causes several problems. For example, we have to change the environment directories in each script we are working with or have to deal with different packages versions incompatibilities as we said before. So, in order to avoid this problem, we think that it will be good to use Docker to store the code in a container and, therefore, make its execution easier in any machine.

# CONSTRAINTS FULFILLMENT

* **Constraint 8:** We organized all the notebook according to their corresponding zone and following the architecture of the data analysis backbone schema:

| **Data Analysis Backbone** | | | |
| --- | --- | --- | --- |
| **Zones** | **Feature Generation Zone** | **Training Zone** | **Validation Zone** |
| **Notebooks** | LoadingData  DataPreparation  FeatureTransformation  FeatureSelection  DataSplit | LoadingData  TrainingModel | ModelValidation  ModelVisualization |
| **Databases** | DB\_FeatureGeneration | DB\_Train&Test | |
| **Other files** |  | Model.pkl | ModelVisualization.pdf |

* **Constraint 9:** We made sure that the results of the built model will not be bias by applying the following two methods:
* **Balancing** the proportion of observations of each class of the target variable (acquired, closed, operating) in both training and testing data. By doing so, we are making sure that there is no predominant class and, therefore, biased results.



**Figure 3:** *Classes proportion of status in training & testing*

* **Applying 5-fold Cross Validation:** with this method, we can ensure if the model we are working with is not biased by the training data. Therefore, we trained/tested the model using 5-fold cross validation. The average accuracy obtained with cross-validation is the same as the accuracy obtained when we divided the dataset with proportion 70-30%, where we validated the model with the testing dataset. Therefore, we can ensure that the data analysis performed is not biased.
* **Constraint 10 & 11:** the fulfillment of this constraint is already explained in the above sections.

# ANALYSIS RESULTS

In this section, we respond to the analytical questions:

* **Which are the factors that most influence when predicting the status of a startup?**

The factors that most influence the prediction are:

* Number of schools in the city where startups are located
* Number of students that passed the math test with problems of English
* Percentage of Asians that passed the math test
* Percentage of the students with economic problems that passed the math test
* The ratio of the teachers versus students in each city
* Funding rounds of a startup
* Grant of a startup
* Venture of a startup
* **If the school quality of a city could be one of the factors that influences the success of the startups in the same city?**
* Yes, the school quality of a city has a considerable impact on the status of a startup. As we commented above, several factors such as the number of students that passed the math test or students with economic factors have an influence in the success of a startup.