Automatic algorithm selection based on input data

The objective of this project is to be able to select a machine learning model from a given list of models that fits best to the data.
The data is a collection of multiple sources, we want to find the best algorithm for each source.

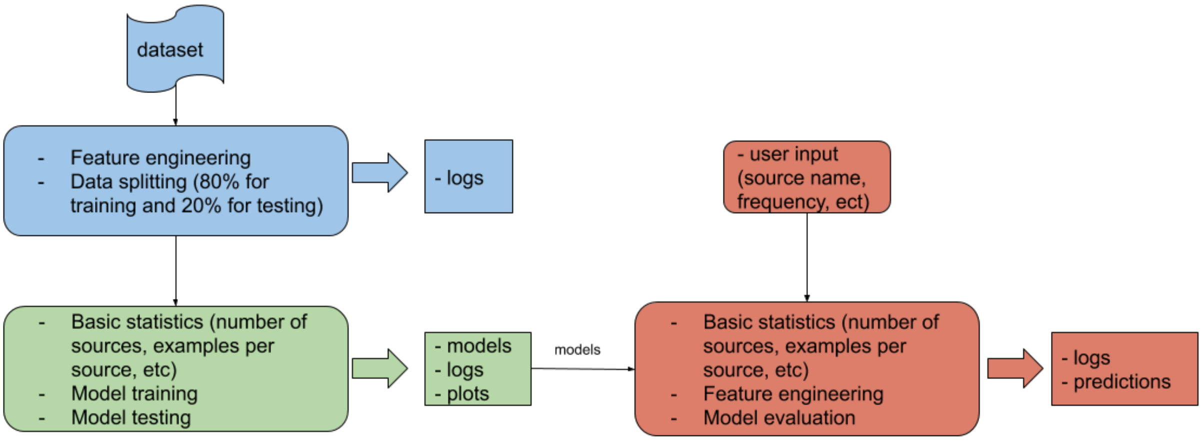
Dataset

The dataset is expected to have, at least, these four fundamental columns:

1. ***id***: the unique identifier for a set of sources.
2. ***source_name***: the algorithm will separate the data into *sources* and try a different machine learning model for each source and will select the best one (according to some metric like recall, precision, f1 or MSE).
3. ***value***: this is the target column, the categories that the algorithm will use to train the models.
4. ***timestamp***: it is assumed that the features are the same for each source, the algorithm will create the following features: *minute*, *hour*, *day*, *month*. These are the features that the algorithm will feed to the machine learning models in order to train them and test them.

System architecture

There are two main pipelines on this project: model training and testing (green) and model evaluation (red).


dataset
- Feature engineering
- Data splitting (80% for training and 20% for testing)
- logs
- user input (source name, frequency, ect)
- Basic statistics (number of sources, examples per source, etc)
- Model training
- Model testing
- models
- logs
- plots
models
- Basic statistics (number of sources, examples per source, etc)
- Feature engineering
- Model evaluation
- logs
- predictions

NOTE: in the rest of the document it is going to be used *<YEAR>/<MONTH>/<DAY>/<YEAR><MONTH><DAY><HOUR><MINUTE><SECOND>*

as <TIMESTAMP>, just for notation purposes.

Pipeline

The pipeline is separated into three major components:

1. **Data preparation (blue):** in this part the features are computed as specified before. Also the data is splitted into training and testing datasets (80% and 20%, respectably). This is done per each source. Sources with less than 50 examples are ignored and saved into *logs/data/<TIMESTAMP>_<FILENAME>_ignored_sources.txt*.
2. **Models training and testing (green):** this component is separated into the following steps:
    a. Statistics for the training and testing data sets are generated and saved into *logs/train/<TIMESTAMP>_<FILENAME>_statistics.json*. Statistics include:
        i. Number of different source names with the number of examples per source name.
        ii. The type of model (regression or classification) for each source name, the way to obtain this is as follows: if the number of unique values represent more than the 30% of total values, then it is assumed that for this source name in particular the model it is going to be regression, if the ratio is less than 30%, then the model will be a classifier.
    b. After that, various machine learning models are going to be trained and the best one is going to be selected for each source. Two outputs are generated:
        i. **Models**: machine learning models for each source, saved into *models/<TIMESTAMP>_<FILENAME>_<source_name>.model*.
        ii. **Logs**: a json file with information regarding to the performance for each model on different sources. This log is located in *logs/train/<TIMESTAMP>_<FILENAME>_models.logs*.
    c. Once the model are trained, they are going to be tested with the remaining 20% of the data per source. Plots of ground truth vs predicted value (if applicable) saved into the file *plots/<TIMESTAMP>_<FILENAME>_<source>.png.*
3. **Model evaluation (red):** in this last step what we want to do is to generate predictions using the models trained in the previous step. The inputs for this component are:
    a. **data file name:** this is the data on which we are basing our predictions.
    b. **path to the models:** the path to the trained models.
    c. **sources:** a comma separated list of the sources we want to predict (if no list is provided then the predictions will run for every source).
    d. **period:** a period of time for generating the timestamps.
    e. **frequency:** the frequency of the data points, measured in minutes (eg, if frequency is 30 minutes, we would have the points 00:00, 00:30, 1:00, 1:30 and so on).

    The output for this step are the following:

a. Statistics saved into *logs/eval/<TIMESTAMP>_<FILENAME>_statistics.json*, which include:
    i. The sources predicted.
    ii. The number of data points.
    iii. Frequency.
    iv. Period.
b. **Predictions:** saved into *output/<TIMESTAMP>_<FILENAME>_predictions.csv*, containing the following columns:
    i. ***source_name***: the source name for the row
    ii. ***id***: the id for the source
    iii. ***timestamp***: the timestamp for predicting the value
    iv. ***prediction***: the predicted value

Machine learning models

Supervised machine learning problems are divided into two categories:

1. Classification: in which a model tries to predict a category from a fixed set of categories (categorical or discrete) given certain input.
2. Regression: in which a model tries to predict a value from a continuous set of numbers.

The models chosen for this project represent the state-of-the-art in classification and regression problems, there is not just one model that can outperform the other, models are usually data-dependant. So for each source, various different models are going to be trained and evaluated, the machine learning models are the following:

1. XGBoost (classification / regression)
2. Support vector machines (classification)
3. Gaussian Naive Bayes (classification)
4. Logistic regression (classification)
5. Linear regression (regression)
6. Lasso (regression)

And the metrics to evaluate each model are going to be: precision, recall, f1 and MSE.

Language and libraries

The language chosen for this project is Python 3.7+. And the main libraries are:

1. Numpy
2. Pandas
3. Scikit-learn
4. Jupyter
5. Seaborn
6. XGBoost

How to run

For training (and testing), run:

```
> python train.py --datapath <path_to_data>
```

Where:

1. *<path_to_data>* is the path to the data.

For evaluating (getting predictions), run:

```
> python eval.py --filename <file_name> --models <path_to_models> --sources <sources>
--period <period> --freq <frequency>
```

Where:

1. *<file_name>* is the name of data file running the predictions on
2. *<path_to_models>* path to the trained models
3. *<sources>* is a list of sources for getting the predictions, separated by commas, eg, “Docks Available,Bikes Available,Commercial Flow - CA” (if no sources are provided, then the predictions will run for every source trained with that dataset)
4. *<period>* specifies the period of time for which the predictions are computed, it is a string with the following format:
    a. *<year_start>,<month_start>,<day_start>-<year_end>,<month_end>,<day_end>*
5. *<frequency>* the distance between data points (measured in minutes)

Example

Running

```
> python eval.py --filename f1 --models models/2019/8/2/ --sources “Docks Avaiable,Bikes
Available” --period “2019,07,22-2019,08,15” --freq 10
```

Will create the following data points for computing the predictions:

Docks Available	id_1	2019-07-25 00:00	1
Docks Available	id_1	2019-07-25 00:10	0
...	...	...	...

Docks Available	id_1	2019-07-26 23:50	12
Bikes Available	id_2	2019-07-25 00:00	‘closed’
Bikes Available	id_2	2019-07-25 00:10	‘open’
...	...	...	...