



Jaqpote5 tutorials

Jaqpote 5: How to deploy a predictive model using the jaqpote library

USE:	How to deploy a predictive model using the jaqpote library
VERSION:	V.2.0 April 2021
CONTACT DETAILS:	Haralambos Sarimveis: hsarimv@central.ntua.gr Pantelis Karatzas: pantelispanka@gmail.com Philip Doganis: filipposd@gmail.com

INTRODUCTION

Jaqpote 5 is a user-friendly web-based e-infrastructure that allows model developers to deploy their predictive models and share them through the web. The Jaqpote 5 GUI directs the model developers to further document their models in a way that can be easily understood and used by end-users with little or no experience on machine learning and statistical analysis. The GUI also allows the end-users to apply the models on their own data for validation and/or prediction purposes and the results are collected and visualised in automatically generated tables, graphs and reports. All major machine learning and statistical data-driven algorithms are supported in Jaqpote 5, by integrating popular libraries such as the Python Scikit-learn and the R Caret libraries. Jaqpote 5 has been designed as a generic modelling and machine learning web platform, but particular emphasis is given on serving the needs of the chemo/bio/nano/pharma/ communities by integrating QSAR, biokinetics, dose-response and read-across models. Jaqpote 5 has been developed by the [Unit of Process Control and Informatics](#) in the School of Chemical Engineering at the National Technical University of Athens.

This document provides a tutorial on how to deploy a model in Jaqpote 5 using the jaqpote library. The resource has been made available at <https://app.jaqpote.org/>. Detailed documentation is available at: <https://www.jaqpote.org/>

DEPLOYING A PREDICTIVE MODEL USING THE JAQPOTPY LIBRARY

Jaqpote 5 is currently integrated with the entire Scikit-learn python library (<https://scikit-learn.org/stable/>) which is the most comprehensive and perhaps the most popular open source library for machine learning, data mining and data analysis. There are plans to integrate algorithms from the R language caret library (<https://cran.r-project.org/web/packages/caret/caret.pdf>) as well as techniques from Julia machine learning libraries, like JuliaML (<https://github.com/JuliaML>).

The main tool developed by NTUA for integrating the Scikit-learn set of algorithms is the jaqpote library, which lets the user create a machine learning model in the python environment of his choice and the deployment the model over the web.

To follow this tutorial the user should:

- 1) Have a Jaqpote 5 account. For more information, please read the tutorial on how to login to Jaqpote 5
- 2) The Jaqpote library should be installed as a pypi package:
`pip install jaqpote`
- 3) One csv file containing the data for a particular modelling example should be available. The file is provided as supplementary material in this deliverable and its name is:
`70_reduced.csv`

After the end of the tutorial, the user should arrive to a web service similar to the one hosted in the following URL: <https://app.jaqpote.org/model/RqCRtRpY85kpbGtsiXp> , which is available to all members of the NanoCommons organisation. For more information about Jaqpote 5 organisations please read the specific tutorial on using and managing organisations in Jaqpote 5.

In the Appendix at the end of this tutorial, the reader can find detailed information about all the functionalities of the Jaqpote 5 library.

The modeller can use the algorithm of her/his choice to fit the best possible model to the available training data and validate the model based on validation statistics. She/he can also use optimal machine learning tools provided in Python, such as TPOT, a Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming (<https://github.com/EpistasisLab/tpot>).

In this tutorial we will demonstrate how to reproduce and publish in Jaqpot5, a Linear nanoQSAR model predicting Solubility of C60 Fullerene in Various Solvents, with just a few lines of code in a Jupyter notebook. The model has been originally presented in the following publication: Farhad Gharagheizi & Reza Fareghi Alamdari (2008) A Molecular-Based Model for Prediction of Solubility of C60 Fullerene in Various Solvents, Fullerenes, Nanotubes, and Carbon Nanostructures, 16:1, 40-57, DOI: 10.1080/15363830701779315.

Import the jaqpotpy library and various components from the pandas and scikit-learn python packages:

```
import pandas as pd

from jaqpotpy import Jaqpot

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, GridSearchCV, RandomizedSearchCV

df=pd.read_csv('70_model_reduced.csv') # Reads the data

print(list(df)) # Prints the headers of all columns
```

Read the data and print the headers of all columns:

```
df=pd.read_csv('70_model_reduced.csv') # Reads the data  
print(list(df)) # Prints the headers of all columns
```

The response is a list with the headers of all columns:

```
['Solvents', 'piPC03', 'ATS1m', 'Seigp', 'More23e', 'H1m', 'logS Exp.']
```

Define the independent variables and the end-point to be predicted and split randomly the dataset into training and test sets consisting of 75% and 25% of the data respectively:

```
Xall=df[['piPC03', 'ATS1m', 'Seigp', 'More23e', 'H1m']] # Defines the columns that will be used as independent features  
  
Yall=df['logS Exp.'] # Defines the end-point  
  
X_train, X_test, Y_train, Y_test = train_test_split(Xall, Yall, train_size=0.75, test_size=0.25, random_state=1)  
# Splits the data into training and test sets
```

Develop a pipeline consisting of scaling the data first and then apply the multiple linear regression algorithm:

```
stepslinear = [('scaler', MinMaxScaler()), ('MLR', LinearRegression())]  
pipelinelinear = Pipeline(stepslinear) # define the pipeline object.
```

Train the model on the training set, compute and print the R^2 statistic on the training set, the test set and on the full set. Finally perform 5-fold cross validation test on the training observation.

```
pipelinelinear.fit(X_train, Y_train)
print('Training score: ', pipelinelinear.score(X_train, Y_train))
print('Testing score: ', pipelinelinear.score(X_test, Y_test))
print('Total score: ', pipelinelinear.score(Xall, Yall))          #Trains the model and prints R^2 statistics

cross_val_score(estimator=pipelinelinear, X=X_train, y=Y_train, cv=5, n_jobs=-1) #Performs a 5-fold cross
validation
```

The response is a list with the calculated statistics:

Training score: 0.8994088488271355

Testing score: 0.9043040438111096

Total score: 0.9034772311898356

array([0.91906039, 0.88995619, 0.90445436, 0.86506266, 0.62316459])

The results are very satisfactory and very similar to the one reported in the paper, where the model has been presented. The following commands are used to deploy the model as a web service in Jaqpot 5. First the user is prompted to enter his username and password:

```
jaqpot = Jaqpot("https://api.jaqpot.org/jaqpot/services/")  
jaqpot.request_key_safe()
```

Only a single command is needed to deploy the model into Jaqpot 5:

```
jaqpot.deploy_sklearn(pipelinelinear,Xall,Yall, title="Linear Model for Predicting Solubility of C60 Fullerenes  
in Various Solvents",description="A description")
```

The response is the unique Jaqpot 5 URL on which the model is hosted.

There are two additional optional arguments in the command which deploys the model in Jaqpot. The first is the definition of the domain of applicability using the leverage method, which is performed by adding the "doa=" argument, followed by the name of a dataframe, which includes only the independent variables. The second argument is the model_meta argument. By setting model_meta=True, a list of metadata is generated and is made available through the Model_meta tab in the Graphical User Interface. This list of metadata contains the algorithm that was used to develop the model, the values of the hyperparameters, all the complete Scikit-learn pipeline, when a series of modelling steps has been used to create the model.

```
jaqpot.deploy_sklearn(pipelinelinear,Xall,Yall, title="Linear Model for Predicting Solubility of C60 Fullerenes  
in Various Solvents",description="A description",doa=Xall, model_meta=True )
```

Optionally, the user can create a Predictive Model Markup Language (PMML) representation of the model to be included as additional information in the web service:

```
!pip install sklearn2pmml

from sklearn2pmml.pipeline import PMMLPipeline

from sklearn2pmml.pipeline import PMMLPipeline
pipelinepmmllinear = PMMLPipeline([
    ("scaler", MinMaxScaler()), ("MLR", LinearRegression())
])
pipelinepmmllinear.fit(X_train, Y_train)

from sklearn2pmml import sklearn2pmml

sklearn2pmml(pipelinepmmllinear, "SolubilityC60linear.pmml", with_repr = True)
```

The model developer can now enter the Jaqpot UI to add any other information about the model (for example detailed description, standard reports like Quantitative Model Reporting Format (QMRF), PMML representations, ontological annotations etc.)

The overview tab: The overview tab opens a markdown free-text editor where the user can include any information about the model. The editing mode can be activated by clicking the red icon on the bottom right of the page (Figure 1) and is deactivated by clicking on the floppy disk icon (Figure 2). The screenshot in Figure 1 is from our full implementation of the model, where we are providing links to the full dataset, the training dataset and the test data sets, a link to a full QMRF report generated using the QMRF editor provided by JRC: <https://sourceforge.net/projects/qmrf/> the report an editable version of the QMRF report and the PMML representation of the model. For uploading dataset to Jaqpot 5 please read the tutorial on uploading datasets.

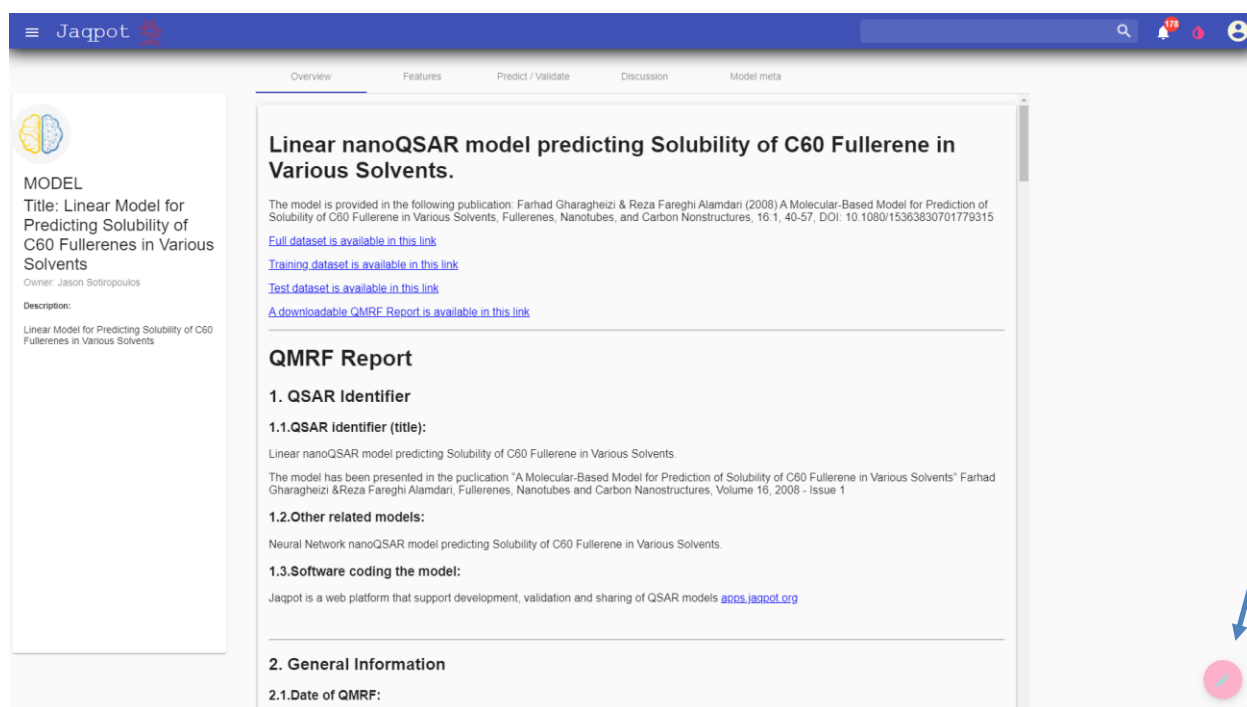


Figure 1. Overview tab (activating the editing mode).

The screenshot displays the Jaqpot web application. The top navigation bar includes the Jaqpot logo, a search bar, and user profile icons. The main interface is divided into a sidebar on the left and a main content area on the right. The sidebar contains a 'MODEL' section with the title 'Linear Model for Predicting Solubility of C60 Fullerenes in Various Solvents', the owner 'Jason Sotiropoulos', and a description. The main content area shows the 'Overview' tab, which contains a code editor with the following text:

```

Edit overview *
# Linear nanoQSAR model predicting Solubility of C60 Fullerene in Various Solvents.

The model is provided in the following publication: Farhad Gharagheizi & Reza Fareghi Alamdari (2008) A Molecular-Based Model for Prediction of Solubility of C60 Fullerene in Various Solvents, Fullerenes, Nanotubes, and Carbon Nanostructures, 16, 1, 40-57, DOI: 10.1080/15363830701779315

[Full dataset is available in this link](https://app.jaqpot.org/dataset/cjXijKX0k6BjelkHMZuNg)
[Training dataset is available in this link](https://app.jaqpot.org/dataset/MYs9MrJuuJmGSjGhwq3DE72)
[Test dataset is available in this link](https://app.jaqpot.org/dataset/RHokDX86xSN3BLj9yL2E5x)

[A downloadable QMRF Report is available in this link](https://github.com/ntua-unit-of-control-and-informatics/QSAR-Models/blob/master/Linear%20Model%20Predicting%20Solubility%20of%20C60%20fullerenes.xml.pdf)

***
# QMRF Report

## 1. QSAR Identifier

### 1.1 QSAR identifier (title):

Linear nanoQSAR model predicting Solubility of C60 Fullerene in Various Solvents.

The model has been presented in the publication "A Molecular-Based Model for Prediction of Solubility of C60 Fullerene in Various Solvents" Farhad Gharagheizi & Reza Fareghi Alamdari, Fullerenes, Nanotubes and Carbon Nanostructures, Volume 16, 2008 - Issue 1

### 1.2 Other related models:

Neural Network nanoQSAR model predicting Solubility of C60 Fullerene in Various Solvents.

### 1.3 Software coding the model:

Jaqpot is a web platform that support development, validation and sharing of QSAR models [apps.jaqpot.org](apps.jaqpot.org)

&nbsp;
&nbsp;

```

In the bottom right corner of the interface, there are three circular icons: a red 'X' icon, a blue 'Y' icon, and a blue 'Z' icon. A blue arrow points to the red 'X' icon, indicating the deactivation of editing mode.

Figure 2. Overview tab (deactivating the editing mode).

The features tab: Here the model creator can provide specific information about the independent features and the end-point of the model: descriptions, units and ontological classes, which will allow the model to understand data sets that are ontologically annotated automatically. Below (Figure 3) we provide a screenshot of the data tab under the C60 solubility model. Like in the overview tab, the user can enter or change all that information by clicking the red icon on the bottom right of the page, which activates the editing mode (Figure 4)

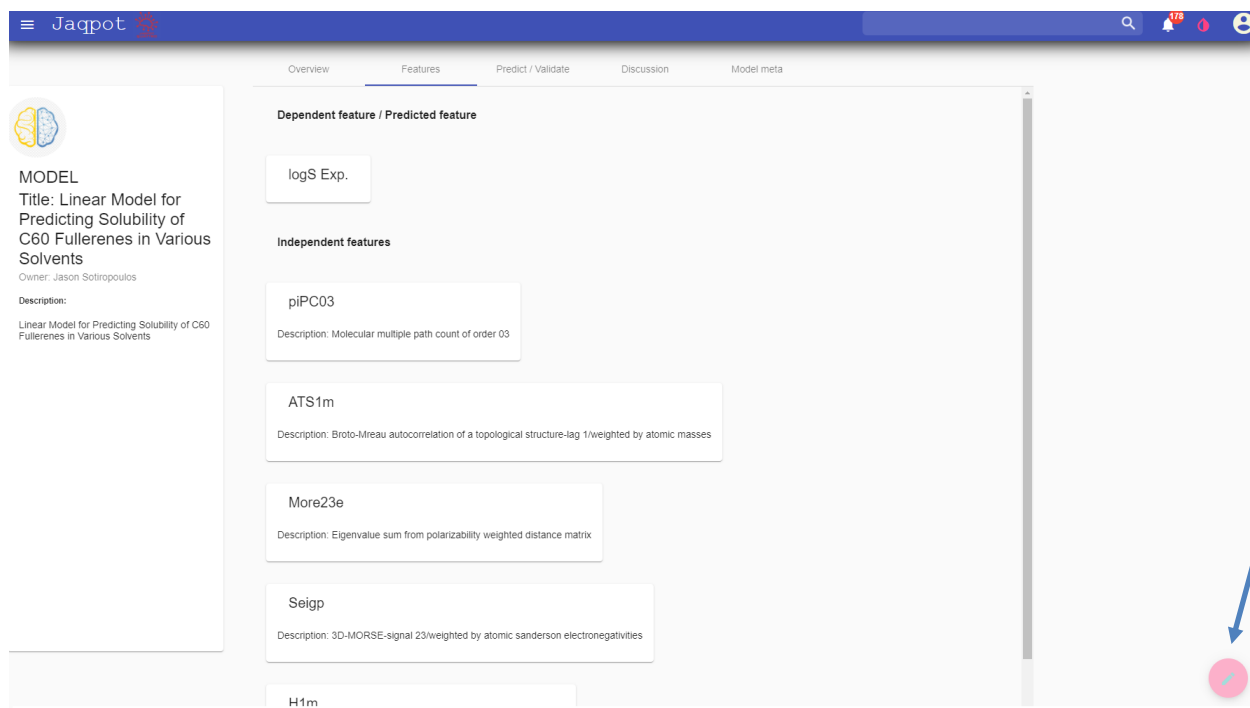


Figure 3. The features tab (entering information and ontological annotations for the independent variables and the end-point predicted by the model).

The screenshot shows the Jaqpot web application interface. The top navigation bar is blue with the Jaqpot logo and a user profile icon. The main content area has a sidebar on the left and a central panel. The sidebar contains a model card for 'Linear Model for Predicting Solubility of C60 Fullerenes in Various Solvents' by user 'hsarimv'. The central panel has tabs for 'Overview', 'Features', 'Predict / Validate', 'Discussion', and 'Archive'. The 'Features' tab is active, showing a form for editing features. The form is divided into two sections: 'Dependent feature / Predicted feature' and 'Independent features'. The 'Dependent feature' section contains a text input for 'logS Exp.' with fields for 'Description', 'Units', and 'Ontological Classes'. The 'Independent features' section contains a text input for 'piPC03' with a 'Description' field. The bottom right corner of the interface has three circular icons: a share icon, a comment icon, and a delete icon.

MODEL
 Title: Linear Model for Predicting Solubility of C60 Fullerenes in Various Solvents
 Owner: hsarimv
 Description
 Linear Model for Predicting Solubility of C60 Fullerenes in Various Solvents

Dependent feature / Predicted feature

logS Exp.

Description

Units

Ontological Classes

Independent features

piPC03

Description
 Molecular multiple path count of order 03

Figure 4. The features tab in editing mode.

The model is now complete. The user can share the model with his partners or the community through the Jaqpot organisations. To do that the user clicks on the share button, which is available in all tabs.

He can choose to share the models in two levels of rights (write, execute) with organisations where he is a member (Figure 5). For more information about Jaqpot 5 organisations please read the specific tutorial on using and managing organisations in Jaqpot 5.

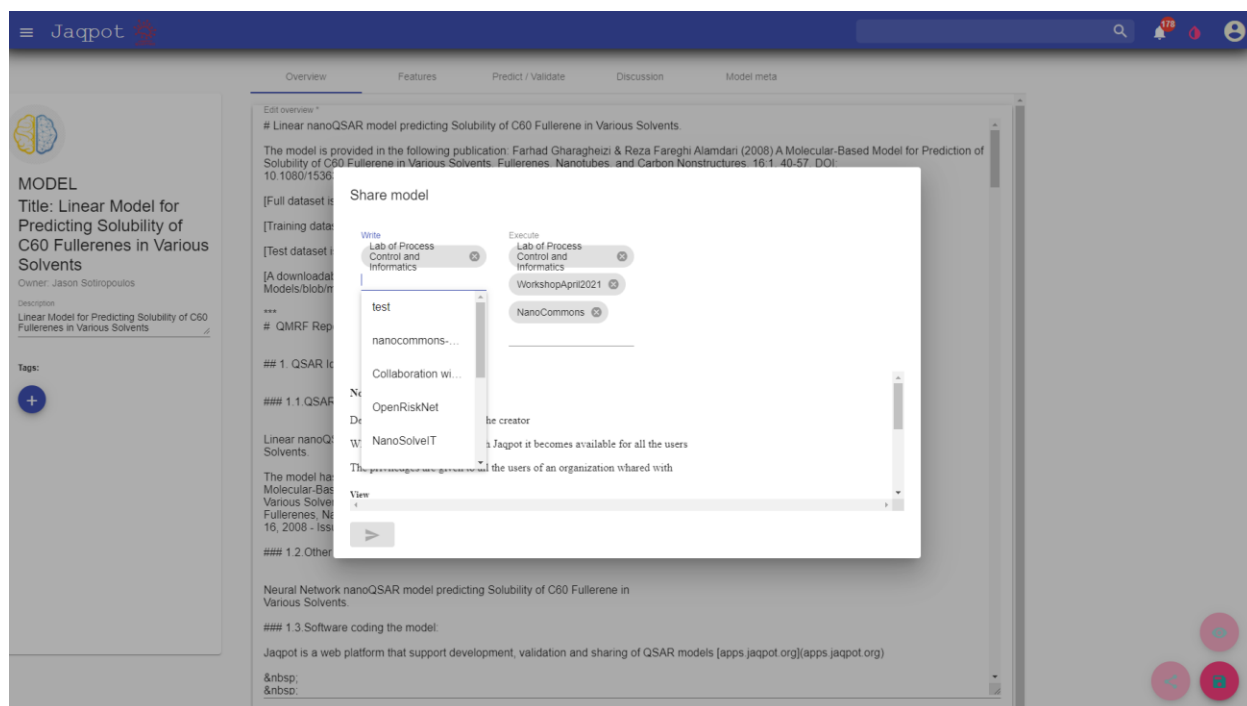


Figure 5. Sharing the model with organisations.

Appendix. Guidelines on installing and using the *jaqpotpy* library

Installation

In order to use jaqpotpy, users need to install it first. Installation can be executed conveniently as a pypi package.

```
pip install jaqpotpy
```

Usage and initialization

Import Jaqpot

After installation, Jaqpot needs to be imported with the following command:

```
from jaqpotpy import Jaqpot
```

Initialize Jaqpotpy on the services where jaqpot lives.

```
jaqpot = Jaqpot("https://api.jaqpot.org/jaqpot/services/")
```

User authentication by Jaqpot

In order to access jaqpot services, first the authentication of the user is required. First it is necessary to define the web location of the Jaqpot instance being used.

```
jaqpot = Jaqpot("https://api.jaqpot.org/jaqpot/services/")
```

Using Jaqpot username/password

The following command will send your username and password, execute your login and set the api key that is needed:

```
jaqpot.request_key('username', 'password')
```

Same as above, this command hides the password if jaqpot is used through a jupyter notebook etc. and initiates a prompt for your username and password, only visible by the user:

```
jaqpot.request_key_safe()
```

Using token (for users accessing Jaqpot with a Google/Github account)

Alternatively, for users that have logged in through Google or Github it is possible to login with the use of an API key. At the account page, the user can find an api key that can be used in order to have access to the services and send it to Jaqpot by substituting the `api_key` field.

```
jaqpot.set_api_key("api_key")
```

Please note that these keys have short life and should be updated on each login.

Deploy your models!

You can use the commands below, customised per model type, in order to make models trained with scikit-learn algorithms available as web services through Jaqpot. Please note:

- all models should be trained with variables that are pandas dataframes
- when calling a `jaqpot.deploy` function you should use exactly the same variables used to train the model
- The Y variable (prediction endpoint) should have an index if it is a supervised method.

1. `deploy_sklearn()`

Lets you deploy models created with scikit-learn with supervised methods.

`deploy_sklearn()` parameters are:

- **model** : {is a sklearn trained model} A trained model that occurs from the `sklearn.linear_model` family of algorithms
- **X** : {is a pandas dataframe} The dataframe that is used to train the model (X variables).
- **y** : {is a pandas dataframe} The dataframe that is used to train the model (y variables).
- **title** : {is a String} The title of the model
- **description** : {is a String} The description of the model
- **doa** : {is a pandas dataframe} The dataframe that is used to train the model (X variables) (Optional by default null).
- **model_meta** : {boolean} True or False (optional by default False).

The id of the model is returned. The model can be found on the Jaqpot homepage of the user for editing / sharing / execution (create predictions).

Example usage

```
from jaqpotpy import Jaqpot
import pandas as pd
```

```
from sklearn import linear_model

df = pd.read_csv('/path/train.csv')
X = df[['Pclass', 'SibSp', 'Parch', 'Fare']]
y = df['Survived']

clf = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial').fit(X, y)

jaqpote.deploy_sklearn(clf, X, y, title="Sklearn 2", description="Logistic regression model from python for the titanic dataset")
```

On the above example a linear model (in our case a logistic regression) is created and deployed on Jaqpote. The dataset is read as a pandas dataframe (a requirement for Jaqpote 5) and the X and y dataframes are created, on which the algorithm is trained and the model is created.

2. deploy_XGBoost()

Lets you deploy models created with XGBoost library.

deploy_XGBoost() parameters are:

- **model** : {is a sklearn trained model} A trained model that occurs from the sklearn.linear_model family of algorithms
- **X** : {is a pandas dataframe} The dataframe that is used to train the model (X variables).
- **y** : {is a pandas dataframe} The dataframe that is used to train the model (y variables).
- **title** : {is a String} The title of the model
- **description** : {is a String} The description of the model
- **doa** : {is a pandas dataframe} The dataframe that is used to train the model (X variables) (Optional by default null).
- **model_meta** : {boolean} True or False (optional by default False).

The id of the model is returned. The model can be found on the Jaqpote homepage of the user for editing / sharing / execution (create predictions).

3. deploy_sklearn_unsupervised()

Lets you deploy models created with unsupervised sklearn methods. These methods do not require Y since the models are created without one.

deploy_XGBoost() parameters are:

- **model** : {is a sklearn trained model} A trained model that occurs from the sklearn.linear_model family of algorithms
- **X** : {is a pandas dataframe} The dataframe that is used to train the model (X variables).
- **title**: {is a String} The title of the model
- **description**: {is a String} The description of the model
- **doa**: {is a pandas dataframe} The dataframe that is used to train the model (X variables) (Optional by default null).
- **model_meta**: {boolean} True or False (optional by default False).

The id of the model is returned. The model can be found on the Jaqpot homepage of the user for editing / sharing / execution (create predictions).



Support

OpenRiskNet
RISK ASSESSMENT E-INFRASTRUCTURE