

Benjamin Guinebertière

Technical Evangelist, Microsoft France
Azure, data insights, machine learning
[@benjguin](https://twitter.com/benjguin) | <http://3-4.fr>



Web App

The screenshot displays the Visual Studio Online Monaco editor interface. The top status bar shows 'Visual Studio Online "Monaco"' and the file path 'flaskdocumentdb'. The user's name 'Benjamin GUINEBERTIERE' is visible on the right, along with settings, help, and profile icons. The left sidebar contains the 'EXPLORE' view, showing a file tree with 'WORKING FILES' and 'WWWROOT'. The 'views.py' file is selected and highlighted in blue. The main editor area shows the code for 'views.py' in the '/FlaskDocumentDB' directory. The code includes a check for 'hasDbLib', an exception handler for path-related errors, and two routes: a home page and a contact page. Both routes use 'render_template' to display HTML pages with dynamic data like the current year, database link, and a message.

```
views.py /FlaskDocumentDB
27 hasDbLib=False
28 except:
29     message='Exception - sys.path=' + str(sys.path) + '\n' + str(platform.architecture()) + '\n'
30     for i in sys.exc_info():
31         message = message + str(i) + '\n'
32     hasDbLib=False
33
34 @app.route('/')
35 @app.route('/home')
36 def home():
37     """Renders the home page."""
38
39     query=dal.getQueryById(request.args.get('q'))
40     data=dal.getDataAsString(query)
41
42     return render_template(
43         'index.html',
44         title='Home Page',
45         year=datetime.now().year,
46         database_link=databaseLink,
47         collection_link=collectionLink,
48         query=query,
49         data=data,
50         message=message,
51     )
52
53 @app.route('/contact')
54 def contact():
55     """Renders the contact page."""
56     return render_template(
57         'contact.html',
58         title='Contact',
59         year=datetime.now().year,
60         message='Your contact page.'
```

Python on HDInsight

Python2.7 is installed by default on HDInsight 3.0 and later clusters. Hive can be used with this version of Python for stream processing (data is passed between Hive and Python using STDOUT/STDIN).

HDInsight also includes Jython, which is a Python implementation written in Java. Pig understands how to talk to Jython without having to resort to streaming, so it's preferable when using Pig.

Hive and Python

Python can be used as a UDF from Hive through the HiveQL **TRANSFORM** statement. For example, the following HiveQL invokes a Python script stored in the **streaming.py** file.

Linux-based HDInsight

```
add file wasb:///streaming.py;

SELECT TRANSFORM (clientid, devicemake, devicemodel)
  USING 'streaming.py' AS
    (clientid string, phoneLabel string, phoneHash string)
FROM hivesampletable
ORDER BY clientid LIMIT 50;
```

Windows-based HDInsight

```
add file wasb:///streaming.py;

SELECT TRANSFORM (clientid, devicemake, devicemodel)
  USING 'D:\Python27\python.exe streaming.py' AS
    (clientid string, phoneLabel string, phoneHash string)
FROM hivesampletable
ORDER BY clientid LIMIT 50;
```

Python Tools for Visual Studio

The screenshot displays the Microsoft Visual Studio IDE with the PythonApplication.py file open. A context menu is visible over the imports section, listing actions such as 'Remove Imports', 'Refactor', 'Go To Definition', 'Find All References', 'Breakpoint', 'Run To Cursor', 'Run Flagged Threads To Cursor', 'Send to Interactive', 'Send to Defining Module', 'Start without Debugging', 'Start with Debugging', 'Cut', 'Copy', 'Paste', 'Outlining', 'Submit', and 'Source Control'. The code in the background includes imports for numpy, matplotlib, and various sklearn modules, followed by a list of classifier names and a function to generate data for testing.

```
print(__doc__)

# Code source: Gaël Varoquaux
# Modified for documentation by Jaques Grobler
# License: BSD 3 clause

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.lda import LDA
from sklearn.qda import QDA

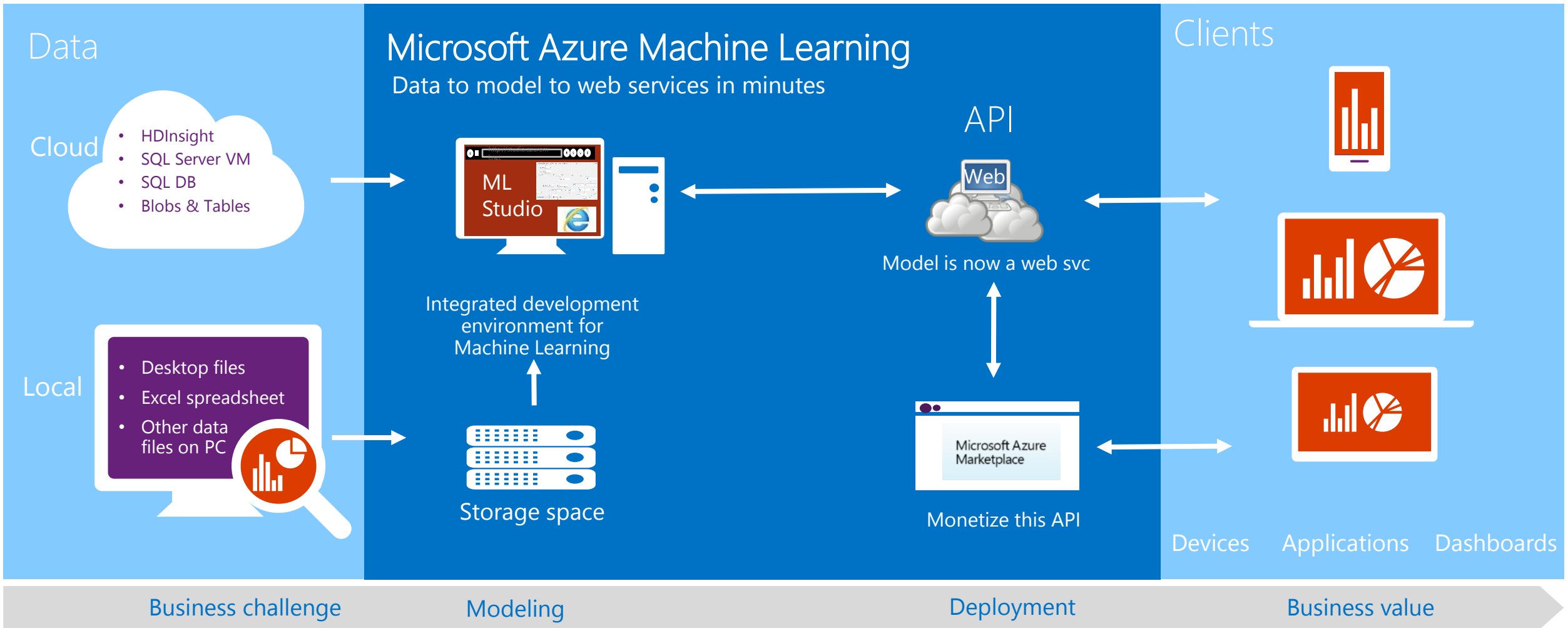
h = .02 # step size in the mesh

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Decision Tree",
         "Random Forest", "AdaBoost", "Naive Bayes", "LDA", "QDA"]
classifiers = [
    KNeighborsClassifier(3),
    SVC(kernel="linear", C=0.025),
    SVC(gamma=2, C=1),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    AdaBoostClassifier(),
    GaussianNB(),
    LDA(),
    QDA()]

X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
                          random_state=1, n_clusters_per_class=1)
rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_separable = (X, y)
```

The Python 64-bit 3.4 Interactive window shows the execution of the code, including the generation of data and the training of classifiers. The Solution Explorer on the right shows the project structure, and the Properties window at the bottom right is empty.

Azure ML





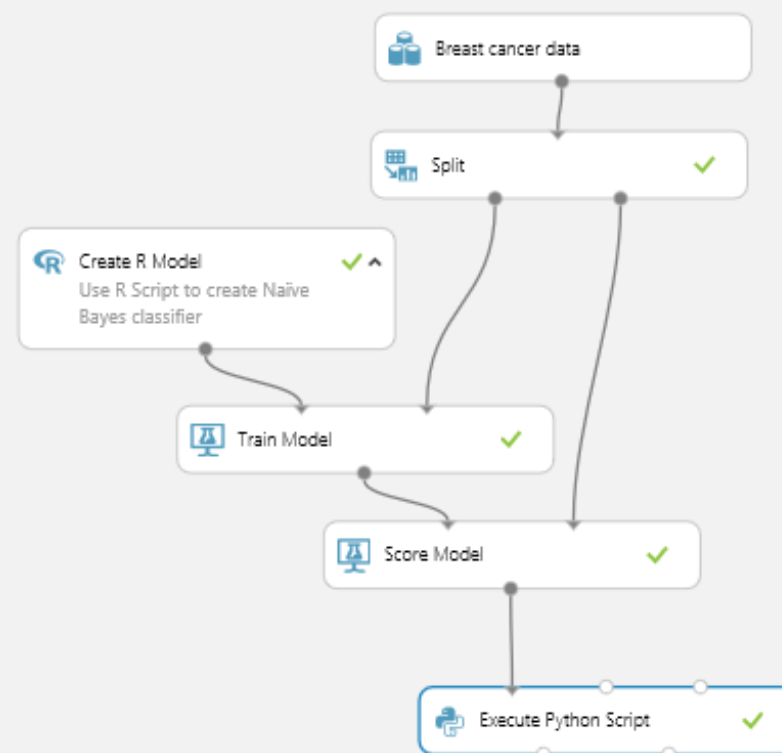
Search experiment items



- ▶ Saved Datasets
- ▶ Trained Models
- ▶ Data Format Conversions
- ▶ Data Input and Output
- ▶ Data Transformation
- ▶ Feature Selection
- ▶ Machine Learning
- ▶ OpenCV Library Modules
- ▶ Python Language Modules
- ▶ R Language Modules
- ▶ Statistical Functions
- ▶ Text Analytics
- ▶ Deprecated
- ▶ Web Service

Sample R and Python

Finished running ✓



Properties

Execute Python Script

Python script

```
1 def azureml_main(dataframe):
2     import matplotlib
3     matplotlib.use("agg")
4
5     from sklearn.metrics import accuracy_score
6     import pandas as pd
7     import numpy as np
8     import matplotlib.pyplot as plt
9
10    scores = dataframe.ix[:, ("Class", "probability")]
11    ytrue = scores["Class"]
12    ypred = np.array([float(val) for val in scores["probability"]])
13    probabilities = scores["probability"]
14
15    accuracy, precision, recall, auc = \
16    accuracy_score(ytrue, ypred), \
17    precision_score(ytrue, ypred), \
18    recall_score(ytrue, ypred), \
19    roc_auc_score(ytrue, probabilities)
20
21    return accuracy, precision, recall, auc
```

START TIME 3/31/2015 11:18:45 AM

END TIME 3/31/2015 11:18:45 AM

ELAPSED TIME 00:00:00

Quick Help



NEW



VIEW RUN HISTORY



SAVE



SAVE AS



DISCARD CHANGES



REFRESH



CANCEL



RUN

PREPARE WEB
SERVICEPUBLISH TO
GALLERYCREATE SCORING
EXPERIMENT

sklearn_plot_classifier_comparison > Execute Python Script > Python device

Standard Output

Standard Error

Interpreter produced no output.

Graphics

