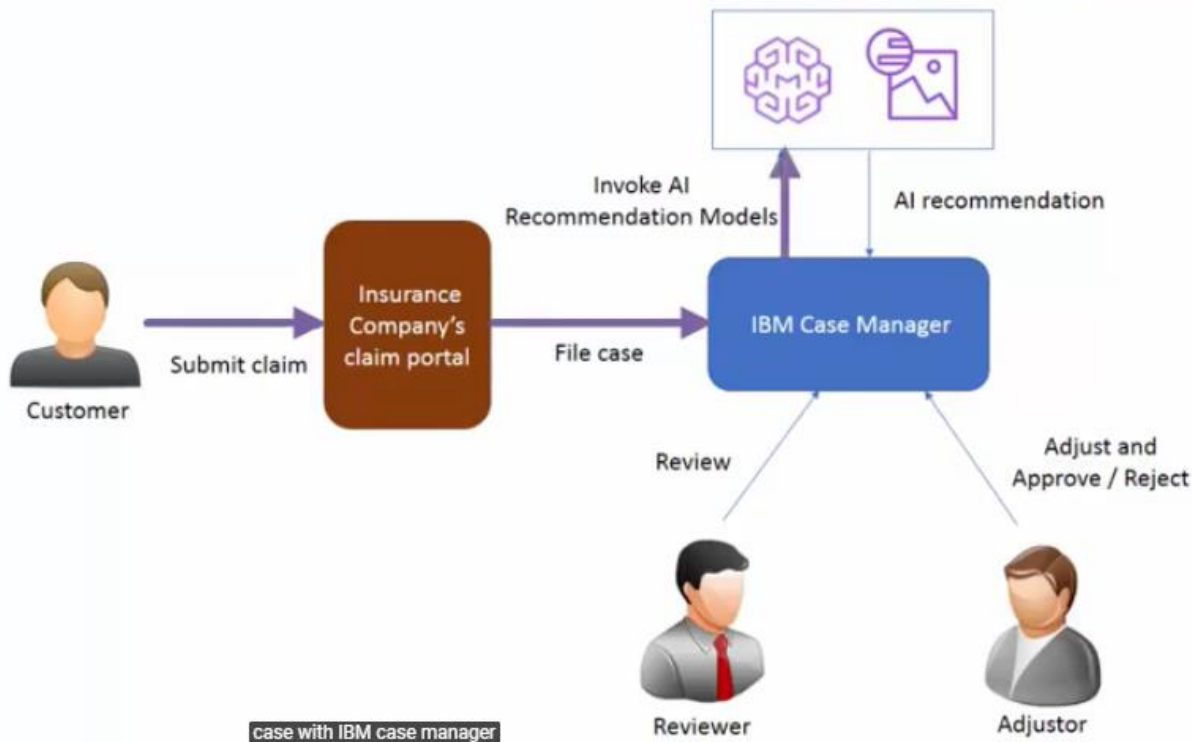


Machine Learning and Visual Recognition models



case with IBM case manager

My Project / Watson Machine Learning

Storage

Type Cloud Object Storage Bucket Name watsonmachinelearning4843016680a48c2bec6b8c21a396aea

0 Byte Used 0% of 5 GB used

Associated services

NAME	SERVICE TYPE	PLAN	ACTIONS
spark-zk	Spark	Personal	...
pm-20-dsx	Machine Learning		...

Access tokens

WMML-access-token

Here you see a project that has all three associated services.

A project all three associated services

GoSales Transactions for Logistic Regression Model

Anonymous outdoor equipment purchase data for machine learning examples.

TOPIC Leisure

LAST UPDATED Dec 20, 2016

CREATE Dec 08, 2016 PUBLISH IBM

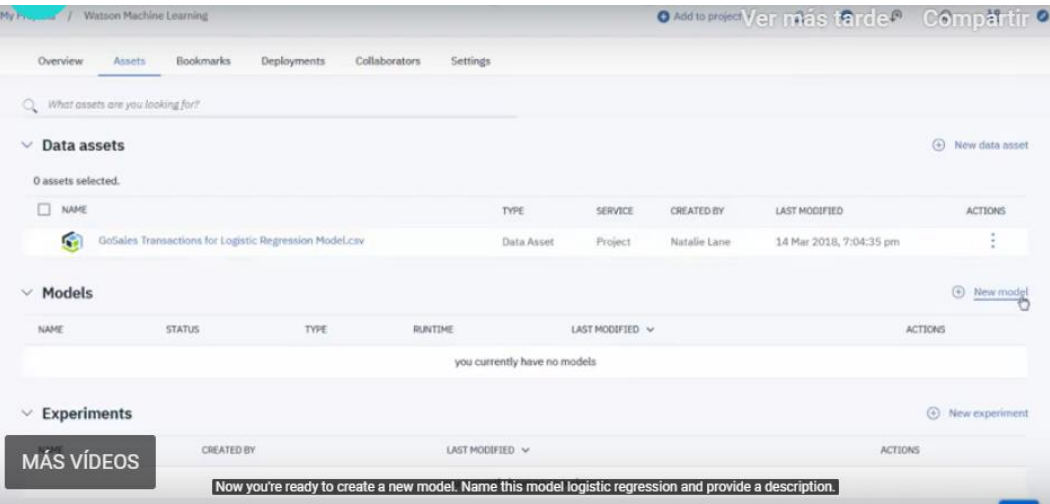
Add to Project

Data Preview

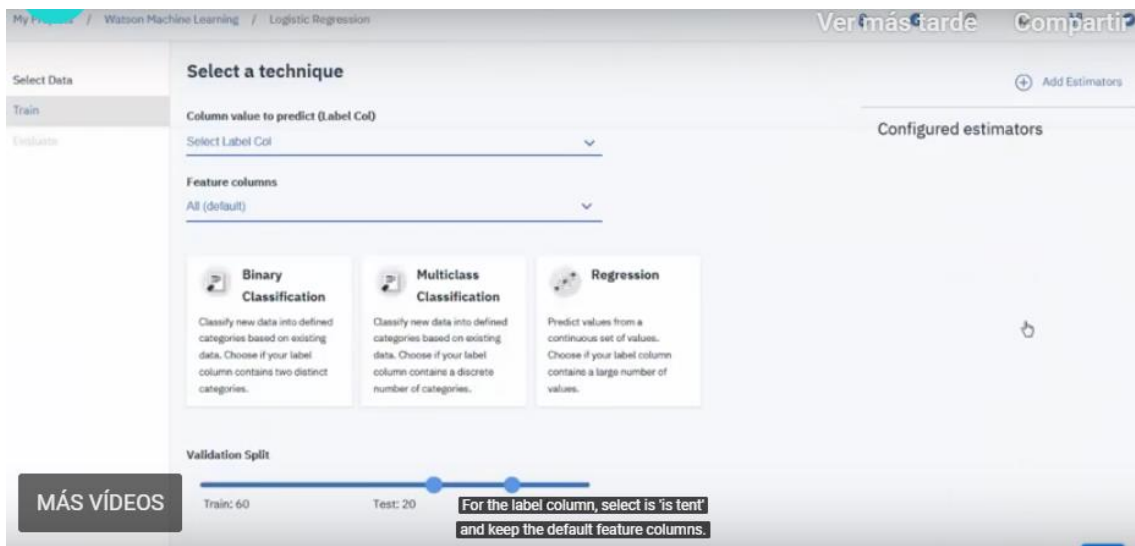
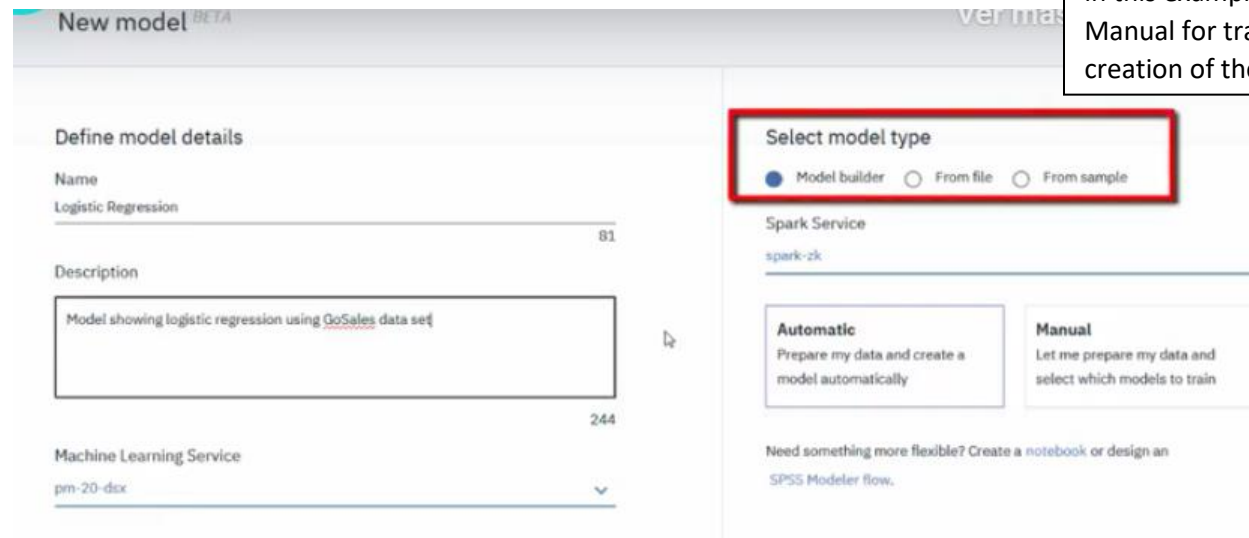
IS_TENT	GENDER	AGE	MARITAL_STATUS	PROFESSION
FALSE	M	27	Single	Professional
FALSE	F	39	Married	Other
FALSE	F	39	Married	Other
FALSE	F	56	Unspecified	Hospitality
FALSE	M	45	Married	Retired
FALSE	M	45	Married	Retired

Add this data set to the Watson Machine Learning project so you can use it later.

You can add the data set to Watson Machine Learning and you can see data preview and column details, in this case add data set of GoSales download since community IBM, late that upgrade the data set you can see in the project and specific in assets



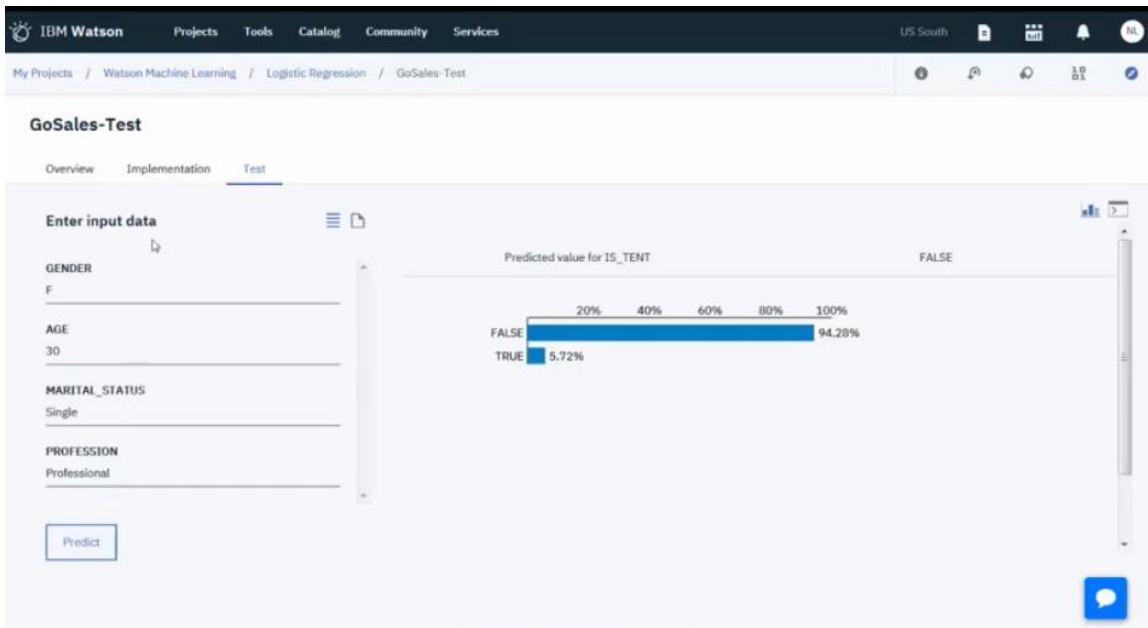
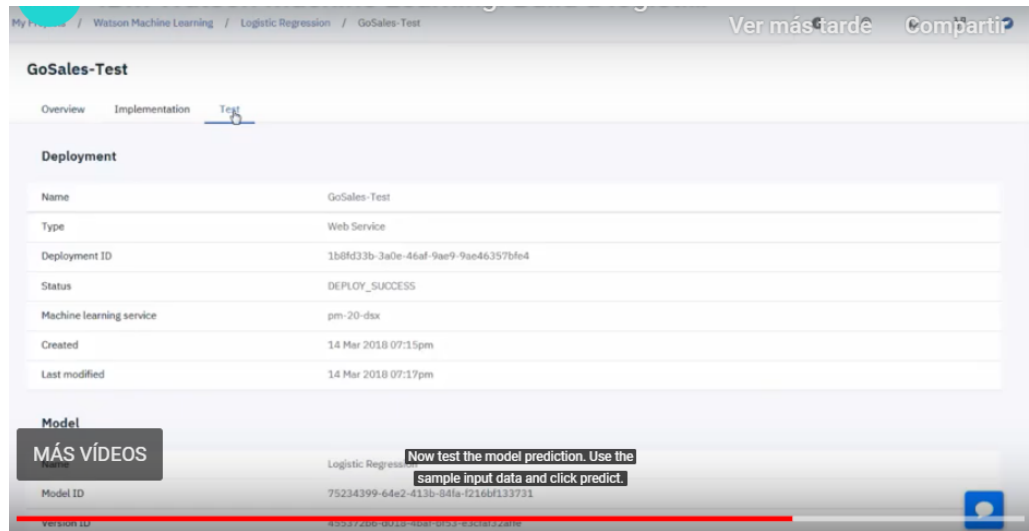
Now for create a new model, you can see that Machine Learning Service and Spark Service are predeterminate and you can create a model since beginning or charger a file with your code or charger a sample model. Also, in this example we choose the option Manual for training method in the creation of the model



In this example we choose the column IS_TENT as label and keep feature columns default, late we selected a binary classification, and add an estimator, in this case we selected Logistic Regression, finally start the process and save the model



We added a deployment for our project, add a name for the deployment and save it, with the deployment save, later we view it and select the option TEST, in this case the deployment used the sample input data, so click in predict, we can change the variable for the prediction



In the Implementation tab we can see the Scoring End-point for future predicts and we can find code snippets for implementation from your application, and finally we can view the API specification. This model is available on the assets tab in the project



IBM Watson Machine Learning: Put a human ...

Putting a human face on machine learning

Create and train a machine learning model in DSX with little to no coding. Save and deploy to Watson Machine Learning on IBM Bluemix and call the model for scoring with a simple Python Flask Web application. A front end will be created to call this deployed model in real time.

TOPIC: Economy & Business

LAST UPDATED: Mar 02, 2018

CREATED: Jul 26, 2017

TERMS OF USE: <https://open-source.org/licenses/ml...>

Copy

Putting a human face on machine learning

IBM Data Science Experience (DSX) is an interactive, collaborative, cloud-based environment where data scientists can use multiple tools to achieve insights. Data scientists can use the best of open source, tap into IBM's unique features, grow their skills, and collaborate with teams. One of the many features of DSX provides the capability to create and train a machine learning model in DSX with little to no coding. This model can subsequently be saved and deployed to Watson Machine Learning on IBM Bluemix and called for scoring in real-time. This tutorial is a continuation of the logistic regression analysis tutorial at <https://data.science.ibm.com/doc/ai/content/analytic-data/ml-example-log-regress.html> which creates, trains, saves and deploys a logistic regression model that predicts the possibility for a tent purchase based on age, sex, marital status and job profession for an individual. In this tutorial, a very simple Python Flask web application front end will be created to call this deployed model in real time.

Table of contents

1. Test model access by using a notebook
 - 1.1 Prerequisites
 - 1.2 Retrieve your credentials
 - 1.3 Authenticate and import libraries
2. Create a user interface
3. Publish app to Bluemix
4. Summary and next steps

Open the notebook and then copy the notebook to your Watson Machine Learning project.

1. Test model access by using a notebook.

The first part of this notebook will confirm that the deployed model can be accessed via an external API. Please note that the credentials and endpoint used in this tutorial are for demonstration purposes only. You must substitute your own credentials and endpoint values.

MÁS VIDEOS

0:45 / 6:04

YouTube

We copy the notebook in Watson Machine Learning project, the first step is to test access created previously in the logistic regression, the prerequisites to exception Python Flask are implement in Cloud, for the service credentials, we find the URL username and password in the service credentials tab and you need also the scoring end-point of your test logistic regression model

```
File Edit View Insert Cell Kernel Help Trusted | Python 2 with Spark 2.1
```

1.3 Authenticate and import libraries.

After you update the following code cells with your specific information, you are ready to run the cells. Run the following cell to define your credentials to the Watson Machine Learning service and to import necessary libraries.

```
In [*]: # Hidden cell
# Copy and paste url, username and password from Watson Machine Learning service credentials.
# Copy scoring endpoint from Data Science Experience deployed model API details.
url = 'https://ibm-watson-ml.mybluemix.net'
username = '4a83192-4d23-441d-87b2-813350e6f087'
password = '52bd6f20-1fa4-400c-a3a9-1ddeaad1bd08'
scoring_endpoint = 'https://ibm-watson-ml.mybluemix.net/v3/wml_instances/c4e072d9-3e5f-4eca-b65f-faac05b92eff/published_models/087d0879-4650-457f-5f-5f'
import urllib, requests, json
```

To retrieve the token that you need to call the scoring API, run the following code:

```
In [2]: # Retrieves token to be used to call scoring API
headers = urllib3.util.make_headers(basic_auth='{}:{}'.format(username, password))
path = '{}/v3/identity/token'.format(url)
response = requests.get(path, headers=headers)
mltoken = json.loads(response.text).get('token')
```

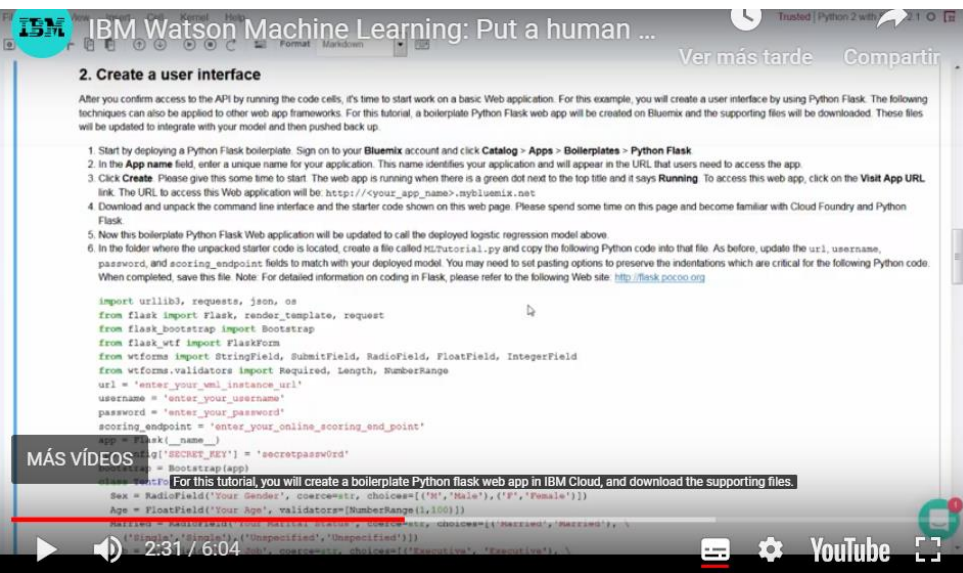
To call the scoring endpoint with some initial sample data, run the following code:

```
In [3]: # Call scoring endpoint with data payload
scoring_header = {'Content-Type': 'application/json', 'Authorization': mltoken}
payload = {'fields': ['GENDER', 'AGE', 'MARITAL_STATUS', 'PROFESSION'], 'values': [['M', 20, 'Single', 'Student']]}
scoring = requests.post(scoring_endpoint, json=payload, headers=scoring_header)
print scoring.text
```

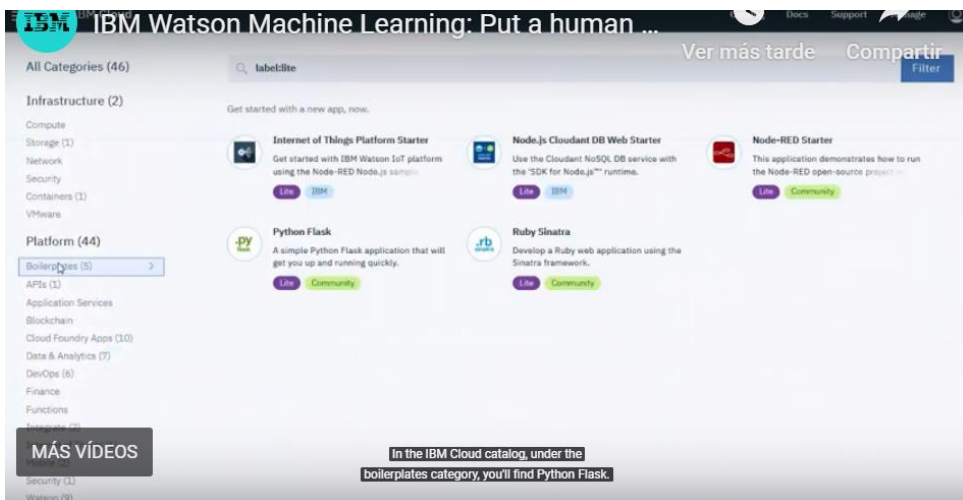
```
{
  "fields": ["GENDER", "AGE", "MARITAL_STATUS", "PROFESSION", "features", "rawPrediction", "probability", "prediction", "nodeADP_class", "nodeADP_classes"],
  "values": [
    [
      "M", 20, "Single", "Student",
      [0.0, 1.9795602967407233, 1.5679400843677074, 2.60755509295228],
      [1.4163195860557454, -1.4163195860557454],
      [0.8047607951511817, 0.19523920484881827],
      0
    ]
  ]
}
```

The next cell retrieves the token that you need to call the scoring API,

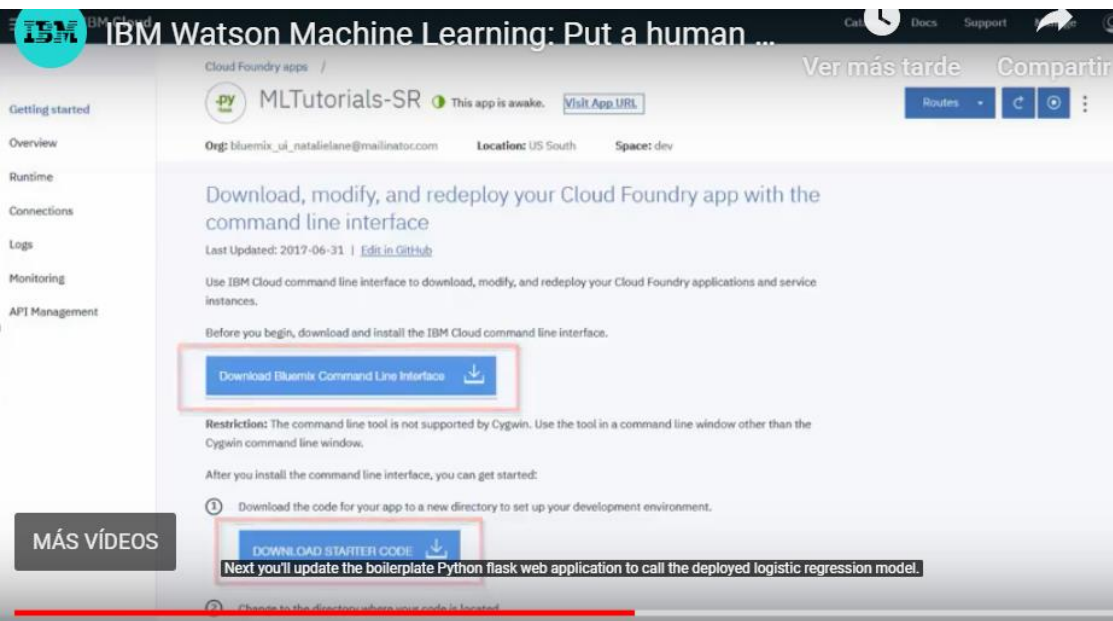
In the first cell we upload our data, in the second cell only is for check the view of our API, and in the third cell we call the scoring end-point with someone initial sample data



For this example, we'll create a boilerplate Python Flask web app in IBM Cloud and download the supporting files, you'll update the files to integrate with your model and then push it back up



In the IBM Cloud catalog, we find boilerplates and we selected Python Flask, here created it, we named it



A time creates the python flask we can see the URL of app and late we downloaded command line interface and starter code, next you'll update the boilerplate python flask, in the folder when you unpack the starter code, created a file called mltutorial.py and copy the python code from this notebook into that file

```
IBM Watson Machine Learning: Put a human ...
MLTutorial.py x Profile x requirements.txt x index.html x score.html
1 import urllib3, requests, json, os
2 from flask import Flask, render_template, request
3 from flask_bootstrap import Bootstrap
4 from flask_wtf import FlaskForm
5 from wtforms import StringField, SubmitField, RadioField, FloatField, IntegerField
6 from wtforms.validators import Required, Length, NumberRange
7 url = 'https://ibm-watson-ml.mybluemix.net'
8 username = '4ae83192-4d23-441d-87b2-813350e6f087'
9 password = '32bd6f20-1fa4-400c-a3a9-1ddeacd1bd08'
10 scoring_endpoint = 'https://ibm-watson-ml.mybluemix.net/v3/wml_instances/c4e072d9-3e5f-4eca-b65f-faac05b92eff/publish'
11 app = Flask(__name__)
12 app.config['SECRET_KEY'] = 'secretpassw0rd'
13 bootstrap = Bootstrap(app)
14 class TentForm(FlaskForm):
15     Sex = RadioField('Your Gender', coerce=str, choices=[('M','Male'),('F','Female')])
16     Age = FloatField('Your Age', validators=[NumberRange(1,100)])
17     Married = RadioField('Your Marital Status', coerce=str, choices=[('Married','Married'),('Single','Single'),('Unspecified','Unspecified')])
18     Job = RadioField('Your Job', coerce=str, choices=[('Executive','Executive'),('Hospitality','Hospitality'),('Other','Other'),('Professional','Professional'),('Retail','Retail'),('Retired','Retired'),('Sales','Sales'),('Student','Student'),('Trade','Trade')])
19     submit = SubmitField('Submit')
20     @app.route('/', methods=['GET','POST'])
21     def index():
22         form = TentForm()
23         if form.validate():
24             url, username, password, scoring_endpoint = form.url, form.username, form.password, form.scoring_endpoint
25             # Update the URL, username, password, and scoring endpoint fields to match your machine learning instance and deployed model.
```

```
MLTutorial.py x Profile
1 web: python MLTutorial.py
```

```
MLTutorial.py x Profile x requirements.txt
1 Flask==0.10.1
2 urllib3
3 requests
4 flask_bootstrap
5 flask
```

Update the URL, username and password and the scoring end-point, saved the file, next is edit profile and change the python file name for Mltutorial.py, late edited the requirements.txt file and paste in the code from the notebook. Now you'll add two files to the template's subfolder, the first is index.html can use for the prediction and the second is score.html displays the results of the prediction

```
MLTutorial.py x Profile x requirements.txt x index.html x score.html
1 {% extends 'bootstrap/base.html' %}
2 {% import "bootstrap/wtf.html" as wtf %}
3 {% block title %}Tent Prediction{% endblock %}
4 {% block navbar %}
5     <nav class="navbar navbar-inverse" role="navigation">
6         <div class="container">
7             <a class="navbar-brand" href="#">Tent Prediction</a>
8         </div>
9     </nav>
10 {% endblock %}
11 {% block content %}
12     <div class="container">
13         <form method="POST" action="">
14             <div class="row">
15                 <div class="col-md-12">
16                     <h3>To determine the probability of a tent purchase, please enter the following:</h3>
17                 </div>
18             </div>
19             <div class="row">
20                 <div class="col-md-8">
21                     <p>{{ form.Sex.label }} {{ wtf.form_field(form.Sex) }}</p>
22                     <p>{{ form.Age.label }} {{ wtf.form_field(form.Age) }}</p>
23                     <p>{{ form.Married.label }} {{ wtf.form_field(form.Married) }}</p>
24                     <p>{{ form.Job.label }} {{ wtf.form_field(form.Job) }}</p>
25                     <p>{{ form.submit }}</p>
26                 </div>
27             </div>
28         </form>
29     </div>
30 {% endblock %}
```

```
C:\MLTutorials-SBR>cf login
API endpoint: https://api.ng.bluemix.net
Email> sharyn.richard@us.ibm.com
Password>
Authenticating...
OK
Targeted org sharyn.richard@us.ibm.com
Targeted space dev
API endpoint: https://api.ng.bluemix.net (API version: 2.75.0)
User: sharyn.richard@us.ibm.com
Org: sharyn.richard@us.ibm.com
Space: dev
C:\MLTutorials-SBR>cf push
Using manifest file C:\MLTutorials-SBR\manifest.yml
Updating app MLTutorials-SBR in org sharyn.richard@us.ibm.com / space dev as sharyn.richard@us.ibm.com
OK
Using route MLTutorials-SBR.mybluemix.net
Uploading MLTutorials-SBR...
Uploading app files from: C:\MLTutorials-SBR
Uploading 23.6K, 16 files
Done uploading
OK
Stopping app MLTutorials-SBR in org sharyn.richard@us.ibm.com / space dev as sharyn.richard@us.ibm.com
OK
```

For run the app, we open the file and cf login, here input the email and password late cf push

Once the app started, we visited the URL and we can check the results

IBM Watson Machine Learning: Put a human ...

Cloud Foundry apps / MLTutorials-SR This app is awake. Visit the URL

Org: bluemix_sa_natalielane@mailinator.com Location: US South Space: dev

Download, modify, and redeploy your Cloud Foundry app with the command line interface

Last Updated: 2017-06-31 | Edit in GitHub

Use IBM Cloud command line interface to download, modify, and redeploy your Cloud Foundry applications and service instances.

Before you begin, download and install the IBM Cloud command line interface.

Download Bluemix Command Line Interface

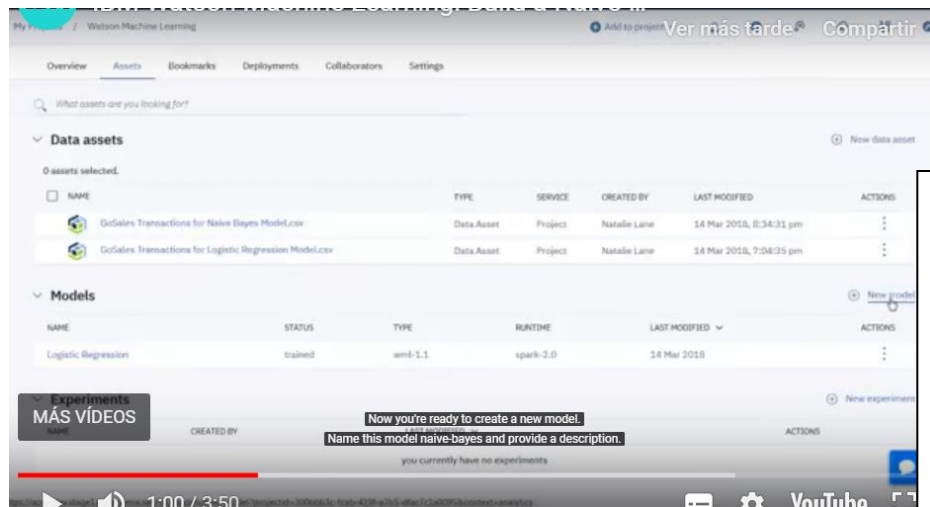
Restrictions: The command line tool is not supported by Cygwin. Use the tool in a command line window other than the Cygwin command line window.

After you install the command line interface, you can get started:

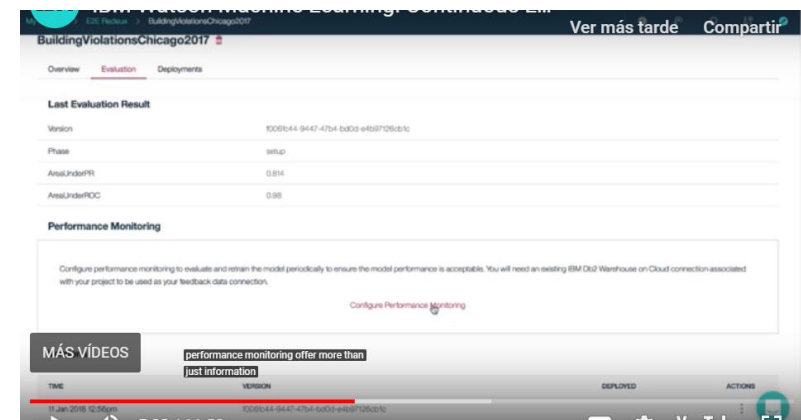
1 Download the code for your app to a new directory to set up your development environment.

2 Once the app is started, visit the URL. Select the criteria and click Submit.

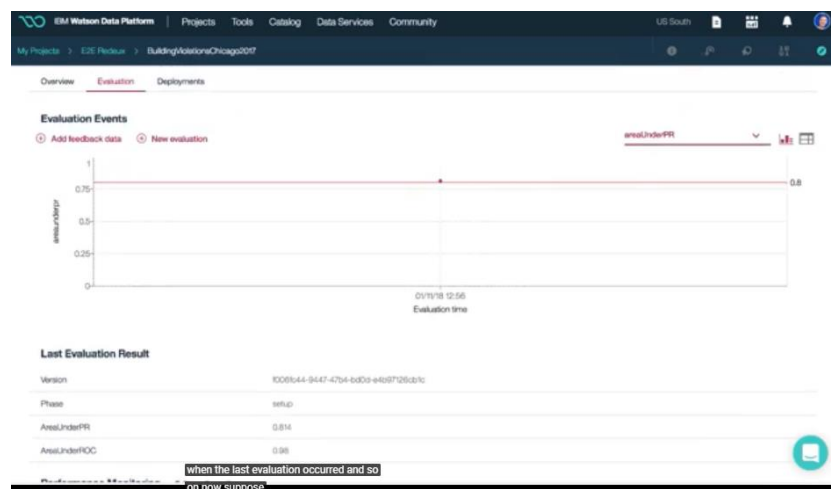
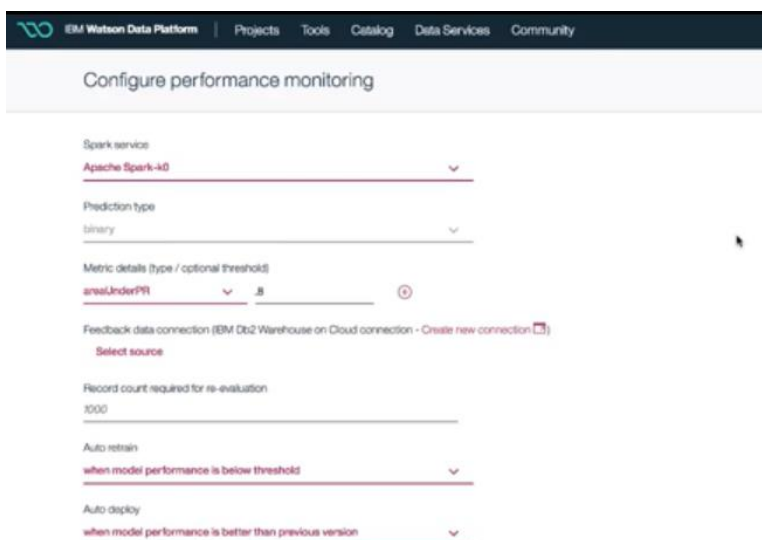
MÁS VÍDEOS

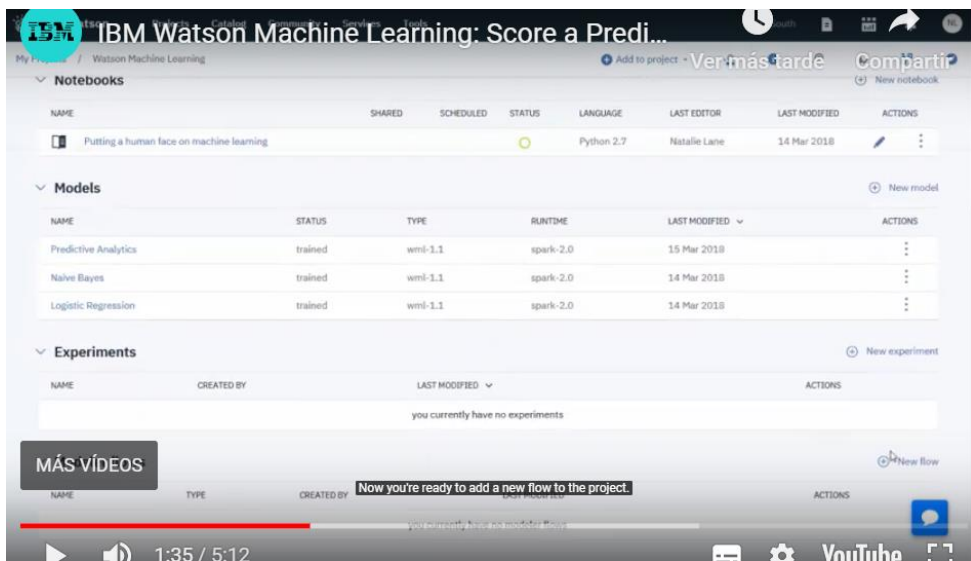


For the Naive-Bayes first add the asset with the name GoSales for Naive-Bayes, late create a new model, in this case created a new model of the way Manual, for the training used the data set GoSales For Naïve-Bayes and in the column value predict used the column PRODUCT_LINE, we used default in feature columns and we selected multiclass classification in estimator selected Naïve Bayes, late train and save the model. In the deployment tab add a new deployment late viewed the deployment

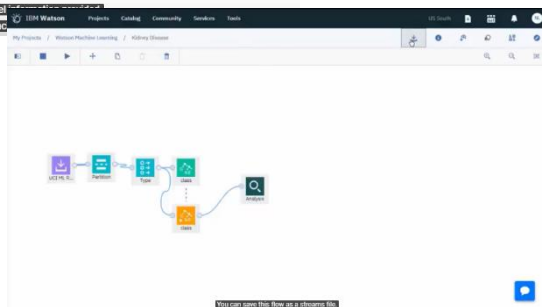
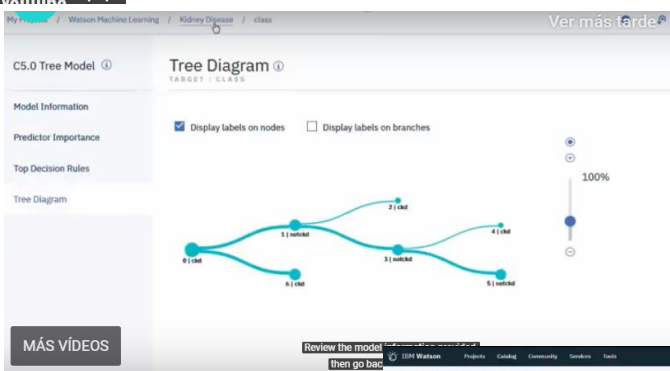
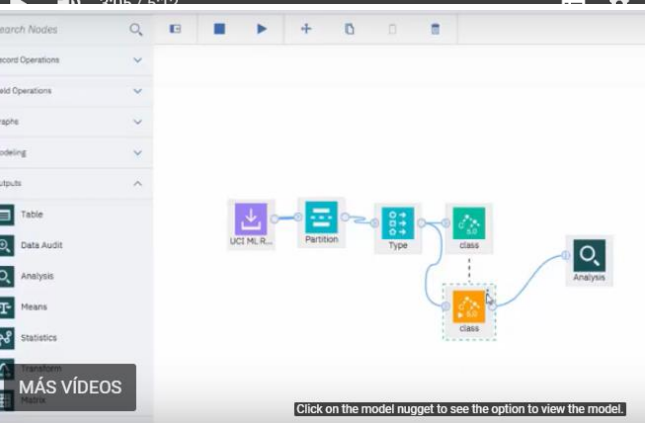
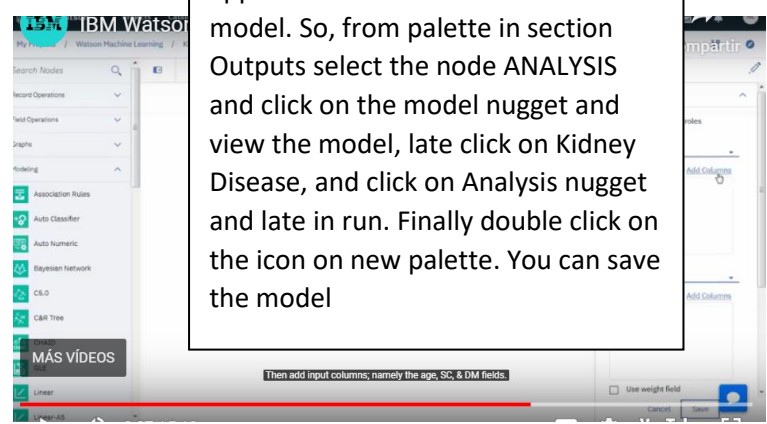
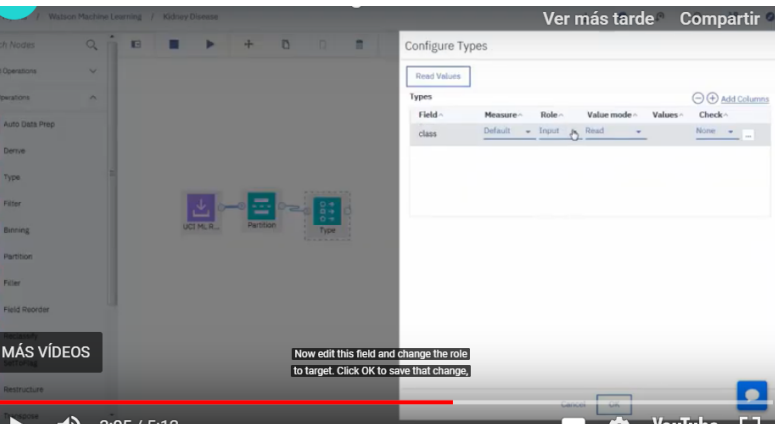
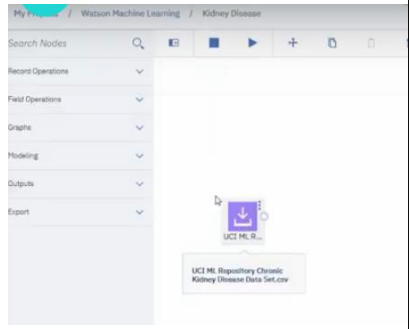
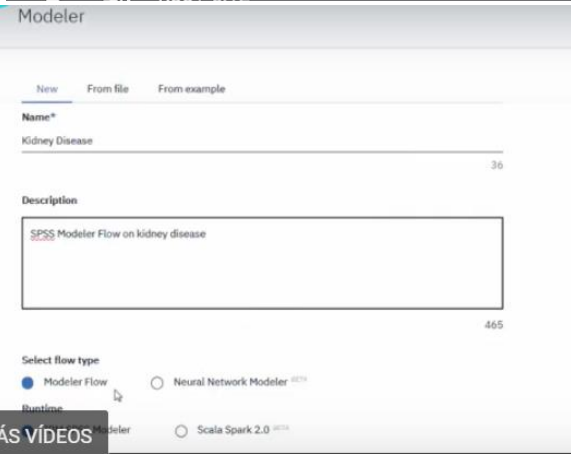


Once created the service into data services, add D2 Warehouse is necessary create the model in this case a logistic regression, for creates this, follow the same steps as you use only Watson Machine Learning, in this example create with two estimators, logistic regression and decision tree. Late in Evaluation tab click in Configure Performance Monitoring, with this option we can see configuration advancer for the evaluation, for example in this case conditional the ROC curve when down over time so retrain the model with new data. In create new connection we put the data base contains the information for our model, and for refresh the information with new data we selected the option SELECT SOURCE and we chose the data base that is refresh, as next step we can charger new data for example in this case the information was about of September but once terminated the period we charged the info of October and the successive. In the DEPLOYMENTS tab we can create a API for do predictions in real time





In this example we used the data set chronic kidney disease, we created new project with three services used and late add a new flow, in this flow selected modeler flow and SPSS IBM Modeler. Once create the model, open for setting our model, here there are our data set, selected the chronic kidney disease. The second node was partition, this for training and test our model. The third node was TYPE, this for create a target variable, in this case CLASS, save the changes. Back on the palette, in the modeling section we chose C5.0, here we selected the field roles CLASS and added columns, the columns are the variable for help to classifier the target value, in this case Age, SC and dm, save the changes and run the model with the icon PLAY. Will appear a new node this is trained the model. So, from palette in section Outputs select the node ANALYSIS and click on the model nugget and view the model, late click on Kidney Disease, and click on Analysis nugget and late in run. Finally double click on the icon on new palette. You can save the model



- w.oosterbosch@nl.ibm.com

- Natasha.Savic@ibm.com

-