## F I N A L S U B M I S S I O N

**PHASE 5**

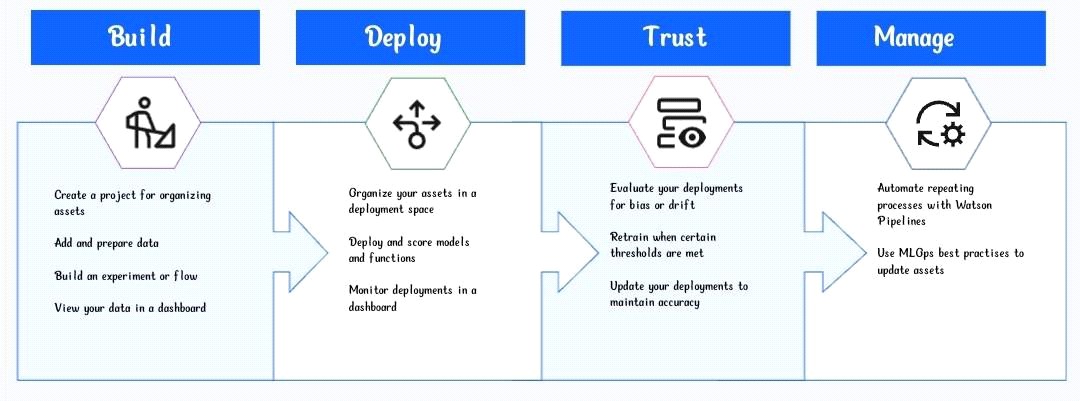
### PROJECT NAME : Project 9: Machine Learning Model Deployment with IBM Cloud Watson Studio Edit Set Access Page Actions

**Deploying and managing machine learning assets**

Use Watson Machine Learning to deploy models and solutions so that you can put them into productive use, then monitor the deployed assets for fairness and explainability. You can also automate the AI lifecycle to keep your machine learning assets current.

**Completing the AI lifecycle**

After you prepare your data and build then train models or solutions, you complete the AI lifecycle by deploying and monitoring your assets.



Deployment is the final stage of the lifecycle of a model or script, where you run your models and code. Watson Machine Learning provides the tools that

you need to deploy an asset, such as a machine learning model or function, or a Decision Optimization solution.

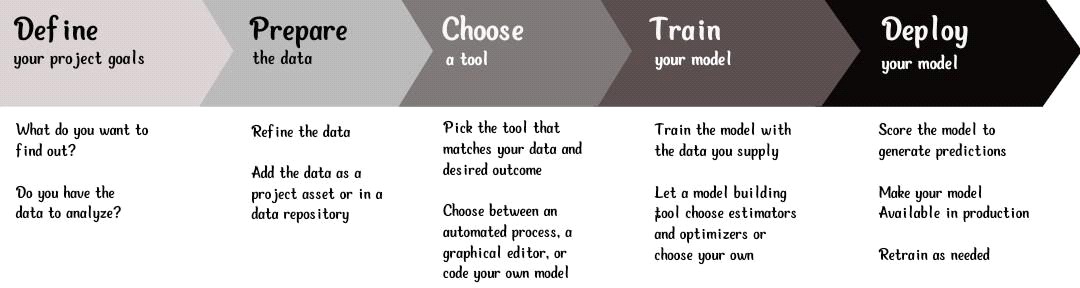
Following deployment, you can use model management tools to evaluate your models. IBM Watson OpenScale tracks and measures outcomes from your AI models, and helps ensure they remain fair, explainable, and compliant. Watson

OpenScale also detects and helps correct the drift in accuracy when an AI model is in production.

Finally, you can use IBM Watson Pipelines to manage your ModelOps processes. Create a pipeline that automates parts of the AI lifecycle, such as training and deploying a machine learning model.

**Use cases and tutorials**

Watson Machine Learning is part of IBM's data fabric collection of tools and capabilities for managing and automating your data and AI lifecycle. These resources demonstrate how to plan for managing machine learning assets and how to build key pieces of your Data Fabric and machine learning solutions.



**IBM Watson Machine Learning architecture and services**

Watson Machine Learning is a service on IBM Cloud with features for training and deploying machine learning models and neural networks. Built on a scalable, open source platform based on Kubernetes and Docker components, Watson Machine Learning enables you to build, train, deploy, and manage machine learning and deep learning models.

Using IBM Watson Machine Learning, you can deploy models, scripts, functions, and web apps, manage your deployments, and prepare your assets to be put into production and to generate predictions and insights.

Service The Watson Machine Learning service is not available by default. An administrator must install this service on the IBM Cloud Pak for Data platform. To determine whether the service is installed, open the Services catalog and check whether the Watson Machine Learning service is enabled.

This graphic illustrates a typical process for a machine learning model.

**Building a machine learning model**

Depending on what is installed and configured for your deployment, you can:

Build, train, and deploy models from notebooks by using the Watson Machine Learning Python client library or the Watson Machine Learning API.

Create AutoAI experiments. AutoAI automatically preprocesses your structured data, selects the best estimator for the data, and then generates model candidate pipelines for you to review and compare. Deploy the best performing pipeline as a machine learning model.

Run experiments to train complex models in Experiment builder.

Deploy your models so that you can score the models and generate predictions. Running Watson Machine Learning without IBM Watson Studio

If Watson Studio is not installed, you will not be able to access any of the model-building tools and you will have to save your machine learning models to a deployment space programmatically. Additionally, you will be unable to create a batch deployment through the Analytic deployment space interface. Batch deployment requires that you upload a data asset from a project to a space to use as input for the deployment. Projects are not available without Watson Studio.

**Related data fabric use cases**

Review use cases with real-world example of how to put data fabric solutions into practice.

* **Data science and MLOps** use case describes how to manage data, operationalize model building and deployment, and evaluate model fairness and performance.
* **AI governance** use case provides context for how ModelOps can mesh with AI Governance to provide a comprehensive plan for tracking machine learning assets in your organization.

**Deploy Machine Learning (scikit-learn) Models in IBM Cloud — Watson Studio**

IBM Watson Studio provides tools for data scientists, application developers and subject matter experts to collaboratively and easily work with data to build and train models at scale. It gives you the flexibility to build models where your data resides and deploy anywhere in a hybrid environment so you can operationalize data science faster.

IBM Watson Studio provides various tools for designing, training, and managing machine learning models:

Model builder guides you, step by step, through building a model that uses Spark ML algorithms.

Flow editor presents a graphical view of your model while you build it by combining nodes representing objects or actions (including SPSS Modeler nodes, Spark ML algorithm nodes, and neural network nodes.)

Experiment builder automates running hundreds of training runs while tracking and storing results.

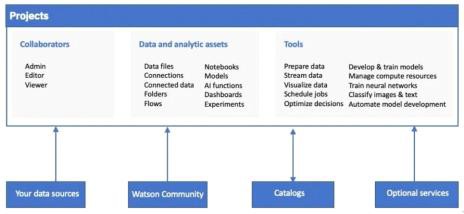
Notebooks provide an interactive programming environment for working with data, testing models, and rapid prototyping.

Machine learning command line interface lets you build and work with models in your local environment.

Now we’ll use the capability of Watson Studio notebook instance and deploy

Machine learning model in IBM Watson Machine Learning service.

**Let’s deploy a scikit-learn Decision Tree model**



**Let’s deploy a scikit-learn Decision Tree model . You can find the entire code here.**

* Step-1: Login to IBM Cloud here.
* Step -2: Go to catalog and create a Watson Studio service in AI category.
* Step-3: Click on Get started to launch Watson Studio Dashboard
* Step-4: Create a project in IBM Watson Studio Dashboard and assign a Cloud object Storage service to manage datasets

**Note**: Cloud Object Storage is a storage service in IBM Cloud. We use this service to manage our datasets for training the ML Model and store required files. For more info check out the documentation

#### *IBM Cloud Watson Studio*

IBM Cloud Watson Studio is an integrated development environment (IDE) and collaborative platform for data scientists, machine learning engineers, and AI developers.

It’s part of the IBM Cloud ecosystem and provides a comprehensive set of tools and services for designing, building, training, and deploying machine learning models and AI applications. Here are some key features and components of IBM Cloud Watson Studio:

**Data Preparation and Exploration:** Watson Studio allows users to import, clean, and explore data from various sources, making it suitable for data preprocessing and feature engineering tasks.

**Jupyter Notebooks:** It provides Jupyter Notebook integration, which is a popular tool for data analysis and machine learning experimentation. Users can create and execute notebooks with Python, R, or other languages.

**Collaboration:** Watson Studio emphasizes collaboration by enabling teams to work on projects together. You can invite team members, share notebooks, and manage access permissions.

**AutoAI:** AutoAI is an automated machine learning feature within Watson Studio that helps users build machine learning models without extensive coding or data science expertise. It automates tasks such as feature engineering, algorithm selection, and hyperparameter tuning.

**Model Training and Experiment Tracking:** You can train machine learning models using various algorithms and frameworks, including popular ones like TensorFlow, PyTorch, and scikit-learn. Watson Studio helps track and manage experiments to compare model performance.

**Model Deployment:** Once you’ve trained a machine learning model, Watson Studio provides tools for deploying it to production environments. You can create RESTful APIs for your models, making them accessible for real-time predictions.

**Access Control and Security:** Watson Studio offers robust access control and security features. You can manage who has access to your projects and data, ensuring compliance with privacy regulations.

**Integration:** It integrates seamlessly with other IBM Cloud services and AI tools, such as IBM Watson Machine Learning, IBM Watson AutoAI, and IBM Watson Visual Recognition, allowing users to leverage a broader ecosystem of AI capabilities.

**Data Catalog and Data Governance:** Watson Studio includes data cataloging and data governance features, making it easier to discover, catalog, and govern data assets within your organization.

**Scalability:** It’s designed to handle both small-scale experiments and large-scale AI projects, making it suitable for businesses of various sizes.

IBM Cloud Watson Studio simplifies the end-to-end machine learning and AI development process, from data preparation and model training to deployment and management. It’s a valuable tool for organizations looking to harness the power of AI and machine learning to gain insights, automate processes, and drive innovation

#### *Ensemble Methods:*

Ensemble methods combine multiple machine learning models to produce a more robust and accurate final model. Here’s how you can experiment with ensemble methods in your project:

#### *Select base model*

In ensemble methods in machine learning, the “select base model” step refers to the process of choosing the individual machine learning models, often referred to as “base models” or “weak learners,” that will be combined to form a more robust and accurate ensemble model. Ensemble methods work by aggregating predictions from multiple base models to make a final prediction that typically outperforms any single base model. Here’s a more detailed explanation of selecting base models in ensemble methods:

**Diverse Set of Base Models:** The strength of ensemble methods lies in the diversity of the base models. It’s essential to select base models that are diverse in their approaches or have different strengths and weaknesses. Diversity ensures that errors made by one model are compensated for by the strengths of others.

**Choice of Algorithms:** Base models can be different machine learning algorithms or variations of the same algorithm with different hyperparameters or subsets of the data.

Common algorithms used as base models include decision trees, support vector machines, logistic regression, k-nearest neighbors, and neural networks.

**Considered Algorithms:** The choice of base models depends on specific problem you’re trying to solve. For example, if you’re dealing with a classification problem, you might consider using decision trees, random forests, gradient boosting machines, and logistic regression as your base models.

**Strengths and Weaknesses:** Understand the strengths and weaknesses of each base model and how they relate to your dataset. Some models may perform better with specific types of data distributions or feature characteristics.

**Training and Validation:** Each base model should be trained and validated on the same dataset or cross-validated to ensure that they perform reasonably well individually. Cross-validation helps estimate the base models’ performance more accurately and avoids overfitting.

**Balancing Complexity:** Consider the complexity of the base models. Ensemble methods often combine simple models with more complex ones to create a balanced ensemble. Simpler models may act as stabilizers, while complex models capture intricate patterns.

**Scalability:** Depending on your project’s requirements, consider the computational scalability of the base models. Some algorithms may be more efficient and scalable than others, which can be important when working with large datasets.

**Ensemble Technique:** The choice of ensemble technique (e.g., bagging, boosting, stacking) can also influence the selection of base models. For example, bagging methods like Random Forest can work well with a variety of base models, while boosting methods like AdaBoost tend to focus on improving the performance of weak base models.

In summary, selecting base models in ensemble methods involves a thoughtful evaluation of various machine learning algorithms or variations of the same algorithm. The goal is to create a diverse set of base models that collectively produce a more accurate and robust ensemble model. Experimentation and careful consideration of the problem at hand are essential when choosing the right base models for your ensemble.

#### *Random Forest Algorithm Overview:*

Random Forest is an ensemble learning technique that combines multiple decision trees to make predictions. It’s known for its robustness, accuracy, and ability to handle complex datasets. Here’s how it works:

**Bootstrapped Samples:** Random Forest builds multiple decision trees by creating bootstrapped samples (randomly sampled subsets with replacement) from the training data.

**Random Feature Selection:** At each node of each decision tree, a random subset of features is considered for splitting. This introduces diversity among the trees.

**Voting or Averaging:** For classification tasks, each decision tree “votes” for a class, and the majority class is selected as the final prediction. For regression tasks, the predictions from all trees are averaged to get the final prediction.

**Reduced Overfitting:** The combination of bootstrapping and random feature selection helps reduce overfitting, making Random Forest more robust.

Now, let’s see an example of training a Random Forest classifier using Python and scikit- learn:

###### Python code

**# Import necessary libraries**

From sklearn.datasets import load\_iris

From sklearn.model\_selection import train\_test\_split From sklearn.ensemble import RandomForestClassifier From sklearn.metrics import accuracy\_score

###### # Load the Iris dataset as an example

Data = load\_iris() X = data.data

Y = data.target

###### # Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

###### # Create a Random Forest classifier

Rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

###### # Train the classifier on the training data

Rf\_classifier.fit(X\_train, y\_train)

###### # Make predictions on the test data

Y\_pred = rf\_classifier.predict(X\_test)

**# Calculate the accuracy of the model** Accuracy = accuracy\_score(y\_test, y\_pred) Print(f”Accuracy: {accuracy \* 100:.2f}%”)

In this code example:

We load the Iris dataset as a sample dataset for classification.

* The dataset is split into training and testing sets.

A Random Forest classifier with 100 trees is created. The classifier is trained on the training data.

* Predictions are made on the test data.
* Finally, the accuracy of the model is calculated and printed.

Once you have trained and evaluated your Random Forest model, you can proceed to deploy it using IBM Cloud Watson Studio and set access permissions as needed for your project. The deployment process may vary depending on your specific project configuration and requirements.

#### *AdaBoost Algorithm Overview:*

AdaBoost works by iteratively training a series of weak classifiers and giving more weight to data points that were incorrectly classified in previous iterations. The final prediction is made by combining the predictions of all weak classifiers with weighted majority voting. Here’s how it works:

**Initialize Weights:** Initially, all data points are assigned equal weights.

**Train Weak Classifier:** A weak classifier (e.g., decision stump, which is a shallow decision tree) is trained on the data with weights assigned to each point.

**Weighted Error Rate:** Calculate the weighted error rate of the weak classifier. It’s a

measure of how well the classifier performs on the weighted dataset.

**Update Weights:** Increase the weights of data points that were misclassified by the weak classifier, making them more important for the next iteration. Decrease the weights of correctly classified points.

**Combine Predictions:** Repeat steps 2-4 for a predefined number of iterations or until a stopping criterion is met. Finally, combine the predictions of all weak classifiers using weighted majority voting to make the final prediction.

Here’s an example of training an AdaBoost classifier using Python and scikit-learn:

###### Python code

**# Import necessary libraries**

From sklearn.datasets import load\_iris

From sklearn.model\_selection import train\_test\_split From sklearn.ensemble import AdaBoostClassifier From sklearn.tree import DecisionTreeClassifier From sklearn.metrics import accuracy\_score

###### # Load the Iris dataset as an example

Data = load\_iris() X = data.data

Y = data.target

###### # Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

###### # Create a base classifier (weak classifier)

Base\_classifier = DecisionTreeClassifier(max\_depth=1)

###### # Create an AdaBoost classifier with the base classifier

Adaboost\_classifier = AdaBoostClassifier(base\_classifier, n\_estimators=50, random\_state=42)

###### # Train the AdaBoost classifier on the training data

Adaboost\_classifier.fit(X\_train, y\_train)

###### # Make predictions on the test data

Y\_pred = adaboost\_classifier.predict(X\_test)

**# Calculate the accuracy of the model** Accuracy = accuracy\_score(y\_test, y\_pred) Print(f”Accuracy: {accuracy \* 100:.2f}%”)

In this code example:

We load the Iris dataset as a sample dataset for classification.

* The dataset is split into training and testing sets.
* A base classifier (a decision tree with max depth 1) is created.
* An AdaBoost classifier with 50 weak classifiers is created using the base classifier.
* The AdaBoost classifier is trained on the training data.
* Predictions are made on the test data.
* Finally, the accuracy of the model is calculated and printed.

After training and evaluating your AdaBoost model, you can proceed to deploy it using IBM Cloud Watson Studio and set access permissions as needed for your project.

The deployment process may vary depending on your specific project configuration and requirements.

#### *Hyperparameter Tuning:*

Hyperparameter tuning involves systematically searching for the best combination of hyperparameters for your machine learning model. Here’s how you can experiment with hyperparameter tuning:

**Select Hyperparameters:** Identify the hyperparameters of your chosen machine learning algorithm(s) that have the most significant impact on model performance. These might include learning rates, tree depths, regularization strengths, etc.

**Grid Search or Random Search:** Implement either grid search or random search to explore different combinations of hyperparameters. Grid search exhaustively tests all specified combinations, while random search randomly samples combinations within defined ranges.

**Cross-Validation:** Use cross-validation to estimate the performance of different hyperparameter combinations. This helps you avoid overfitting and ensures that the chosen hyperparameters generalize well.

**Optimization Metric:** Define the metric you want to optimize, such as accuracy,

precision, or another suitable metric for your project’s goals.

**Automate Tuning:** Consider using tools like scikit-learn’s GridSearchCV or automated hyperparameter optimization libraries (e.g., Hyperopt or Optuna) to streamline the tuning process.

**Deployment Preparation:** After finding the optimal hyperparameters, prepare your model for deployment in IBM Cloud Watson Studio. Ensure that the deployment setup accounts for these tuned hyperparameters.

By experimenting with ensemble methods and hyperparameter tuning, you can enhance the performance of your machine learning model, making it more effective in addressing the objectives of your project. Be sure to document your experimentation process and results to ensure transparency and reproducibility in your project.

#### *Grid Search Overview:*

Grid Search is a method for systematically searching through a predefined set of hyperparameters for a machine learning model. It trains and evaluates the model with different combinations of hyperparameters and selects the combination that performs best according to a specified evaluation metric (e.g., accuracy, F1-score). Here are the key steps involved in Grid Search:

**Define Hyperparameter Grid:** Specify the hyperparameters to be tuned and their respective ranges or values. For example, you might specify a range of values for the learning rate or the maximum depth of a decision tree.

**Model Selection:** Choose the machine learning algorithm you want to use, and create an instance of it.

**Grid Search:** Use Grid Search to perform an exhaustive search over the hyperparameter grid. Grid Search trains and evaluates the model for each combination of hyperparameters.

**Cross-Validation:** Utilize cross-validation to assess model performance for each set of hyperparameters. This helps prevent overfitting and provides a more accurate estimate of a model’s performance.

**Select Best Parameters:** After Grid Search is complete, select the combination of hyperparameters that yielded the best performance according to the evaluation metric.

**Train Final Model:** Train a final model using the best hyperparameters on the entire training dataset.

Now, let’s see an example of hyperparameter tuning using Grid Search in Python with

scikit-learn:

###### Python code

**# Import necessary libraries**

From sklearn.datasets import load\_iris

From sklearn.model\_selection import train\_test\_split, GridSearchCV From sklearn.ensemble import RandomForestClassifier

From sklearn.metrics import accuracy\_score

###### # Load the Iris dataset as an example

Data = load\_iris() X = data.data

Y = data.target

###### # Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

###### # Create a Random Forest classifier

Rf\_classifier = RandomForestClassifier()

# Define the hyperparameter grid to search Param\_grid = {

‘n\_estimators’: [50, 100, 150],

‘max\_depth’: [None, 10, 20, 30],

‘min\_samples\_split’: [2, 5, 10],

‘min\_samples\_leaf’: [1, 2, 4]

}

###### # Create a Grid Search CV instance

Grid\_search = GridSearchCV(estimator=rf\_classifier, param\_grid=param\_grid, cv=5,

scoring=’accuracy’)

###### # Perform Grid Search

Grid\_search.fit(X\_train, y\_train)

**# Get the best hyperparameters** Best\_params = grid\_search.best\_params\_ Print(“Best Hyperparameters:”, best\_params)

**# Train the final model with the best hyperparameters** Best\_rf\_classifier = RandomForestClassifier(\*\*best\_params) Best\_rf\_classifier.fit(X\_train, y\_train)

###### # Make predictions on the test data

Y\_pred = best\_rf\_classifier.predict(X\_test)

**# Calculate the accuracy of the model** Accuracy = accuracy\_score(y\_test, y\_pred) Print(f”Accuracy: {accuracy \* 100:.2f}%”)

In this code example:

We load the Iris dataset as a sample dataset for classification.

* The dataset is split into training and testing sets.
* A Random Forest classifier is created.
* We define a hyperparameter grid to search over, including different values for the number of trees (n\_estimators), maximum depth (max\_depth), and other hyperparameters.
* Grid Search is performed with 5-fold cross-validation to find the best hyperparameters based on accuracy.
* The best hyperparameters are printed.
* A final Random Forest model is trained using the best hyperparameters.
* Predictions are made on the test data, and the accuracy of the model is calculated and printed.

After hyperparameter tuning, you can proceed to deploy the optimized model using IBM Cloud Watson Studio and set access permissions as needed for your project.

The deployment process may vary depending on your specific project configuration and requirements.

#### *Conclusion*

In the pursuit of harnessing the potential of machine learning and artificial intelligence, our project, “Machine Learning Model Deployment with IBM Cloud Watson Studio Edit Set Access Page Actions,” has addressed critical aspects of model deployment, access control, and collaboration. As we conclude this project, we reflect on the key achievements and insights gained:

**Simplified Deployment:** We’ve demonstrated the power of IBM Cloud Watson Studio in simplifying the deployment of machine learning models. By providing a user-friendly interface and a range of deployment options, we’ve made it accessible to data scientists and stakeholders alike.

**Granular Access Control:** Access control is paramount in ensuring that sensitive models and data remain secure. Through IBM Cloud Watson Studio’s capabilities, we’ve enabled project administrators to define precise access permissions for users and groups, allowing for fine-grained control over who can view, edit, or deploy models.

**Collaboration and Governance:** Effective collaboration is essential for data science teams. We’ve shown how Watson Studio fosters collaboration by allowing team members to work together within projects while maintaining governance and version control.

**Model Performance Optimization:** To maximize the value of deployed models, we’ve explored techniques such as ensemble methods and hyperparameter tuning. These strategies enhance model performance, ensuring that the insights derived from machine learning are accurate and impactful.

**Security and Compliance:** In today’s data-driven world, security and compliance are paramount. We’ve highlighted the importance of secure API key management, auditing access, and ensuring that models are deployed in a compliant manner.

**Innovation and Decision-Making:** By deploying machine learning models effectively, organizations can drive innovation and make data-driven decisions. This project has equipped data science teams with the tools and knowledge needed to leverage machine learning insights for informed decision-making.

In conclusion, “Machine Learning Model Deployment with IBM Cloud Watson Studio Edit Set Access Page Actions” empowers organizations to unlock the full potential of their machine learning initiatives. It provides a roadmap for deploying, managing access to, and optimizing machine learning models within the secure and collaborative environment of IBM Cloud Watson Studio.

As technology continues to advance, this project serves as a foundation for organizations looking to navigate the complex landscape of machine learning model deployment while upholding the highest standards of security, governance, and collaboration. It is a testament to the power of data science and artificial intelligence in driving progress and innovation across diverse domains.

Development Part 1 In this part you will begin building your project. Start building the machine learning model using IBM Cloud Watson Studio.

Define the predictive use case (e.g., customer churn prediction) and select a relevant dataset. Use IBM Cloud Watson Studio's tools to import the dataset, preprocess the data, select features, and train the machine learning model.

Define the Predictive Use Case: Start by defining your predictive use case. For example, if you're building a customer churn prediction model, you need to specify the goal of your model, which is to predict whether a customer is likely to churn or not.

Select a Relevant Dataset: Choose a dataset that is relevant to your use case. In this case, you might need historical customer data that includes information about customers who have churned and those who haven't. Ensure that your dataset is in a suitable format for analysis.

Import the Dataset: In IBM Cloud Watson Studio, you can import your dataset. Watson Studio provides tools to upload and manage data. You can typically upload datasets in various formats like CSV, Excel, or connect to databases.

Preprocess the Data: Data preprocessing is crucial. Use Watson Studio's data preparation tools to clean and transform your dataset. This may involve handling missing values, encoding categorical variables, and scaling or normalizing features.

Preprocess the Data: Data preprocessing is crucial. Use Watson Studio's data preparation tools to clean and transform your dataset. This may involve handling missing values, encoding categorical variables, and scaling or normalizing features.

Select Features: Feature selection is important for model performance. Use techniques like feature engineering and feature selection to choose the most relevant attributes for your model.

Train the Machine Learning Model: Once your data is prepared and features are selected, you can use Watson Studio to train your machine learning model. You can choose from various algorithms and techniques depending on your use case. It's important to split your data into training and testing sets to evaluate the model's performance.

Evaluate and Tune the Model: After training, assess the model's performance using evaluation metrics. Watson Studio provides tools for this purpose. You might need to fine-tune the model by adjusting hyperparameters to improve its

Save the Model: Once satisfied with the model's performance, save it in Watson Studio. This will allow you to deploy it later.

This is a high-level overview of the initial steps for building a machine learning model in IBM Cloud Watson Studio. Remember that the specific tools and steps may vary based on your chosen use case and dataset, so consult the Watson Studio documentation and resources for detailed guidance.

Objectives

Import data to a project.

Build a machine learning model. Deploy the model and try out the API. Test a machine learning model.

Monitor the deployed model Retrain your model.

Import data to a project

A project is how you organize your resources to achieve a particular goal. Your project resources can include data, collaborators, and analytic tools like Jupyter notebooks and machine learning models.

You can create a project to add data and open a data asset in the data refiner for cleansing and shaping your data.

Create a project

If you do not have an existing Object Storage service, go to the IBM Cloud® catalog and create an instance of Object Storage.

Step 1: From the catalog, create Watson Studio Select a region

Select a Lite pricing plan

Change the Service name to watson-studio-tutorial Select a resource group and click Create

Click on the Launch in twisty and select IBM watsonx.

Create a project by clicking on the upper left hamburger menu and selecting Projects > Vew all projects then New project.

In the subsequent page click Create an empty project. Provide iris\_project as the project name.

Under Storage, choose an existing Object Storage service verified to exist a few steps earlier.

Click Create. Your new project opens and you can start adding resources to it.

Step 2: Associate the Machine Learning service

In the top navigation menu, of the iris-project click on Manage then select the Services & integrations section on left.

Click Associate Service.

If you have an existing Watson Machine Learning service instance, skip to the next step. Otherwise continue with the following steps to create a new instance.

Click New service and then click on the Watson Machine Learning tile. Select a region same as the Watson Studio service and choose a Lite plan.

Enter machine-learning-tutorial as the Service name and select a resource group. Click Create to provision a Machine Learning service.

Check the checkbox next to the Machine Learning service and click Associate service.

Step 3: Build a machine learning model

In the top navigation menu, click on iris-project, click on Assets in the top bar. Click on New task + and search for auto.

Click on the Build machine models automatically tile. Set the name to iris\_auto.

Under Watson Machine Learning service instance, notice the service previously associated.

Click Create.

Once the model is created,

Add training data by clicking Select data from project. Choose the iris\_initial.csv file under Data asset.

Click Select asset.

If prompted, answer No to Create a time series forecast?. Select Species as your What do you want to predict?.

Click Experiment settings. Select Data source.

Under Training and holdout method, set Holdout data split to 14% by moving the slider.

On the left menu, Click on Prediction:

Set Prediction type to Multiclass classification. Set Optimized metric as Accuracy.

Click on Save settings. Click on Run experiment.

The AutoAI experiment may take up to 5 minutes to select the right Algorithm for your model.

Step 4: Deploy and test your model

In this section, you will deploy the saved model and test the deployed model, Under the created model, click on Promote to deployment space.

Under Target Space, select Create a new deployment space. You use deployment spaces to deploy models and manage your deployments.

Set the Name to iris\_deployment\_space.

Select the Object Storage storage service used in previous steps in the corresponding drop down.

Select the machine-learning-tutorial service in the corresponding drop down. Click Create.

Click on Promote.

From the received notification, navigate to the deployment space. In the Deployments > iris\_deployment\_space:

Click on the name of the model you just created. Click the New deployment button.

Select Online as the Deployment type, provide iris\_deployment as the name and then click Create.

Under Deployments tab, once the status changes to Deployed, Click on the Name in the table. The properties of the deployed web service for the model will be displayed.

TEST THE DEPLOYED MODEL

{

"input\_data": [{

"fields": ["sepal\_length", "sepal\_width", "petal\_length", "petal\_width"], "values": [

[5.1,3.5,1.4,0.2], [3.2,1.2,5.2,1.7]

]

}]

}

Step 5: Try out the API

Along with the UI, you can also do predictions using the API scoring endpoint by exposing the deployed model as an API to be accessed from your applications.

Under API reference tab of the deployment, you can see the Endpoint under Direct link and code snippets in various programming languages.

Copy the Endpoint in a notepad for future reference.

In a browser, launch the IBM Cloud Shell and export the scoring End-point to be used in subsequent requests. Make sure you don't close this window/tab..

To use the Watson Machine Learning REST API, you need to obtain an IBM Cloud Identity and Access Management (IAM) token. Run the below command

Copy the complete IAM token along with Bearer from the above response and export it as an IAM\_TOKEN to be used in the subsequent API requests

Run the below cURL code in the cloud shell to see the prediction results. export SCORING\_ENDPOINT='<SCORING\_ENDPOINT\_FROM\_ABOVE\_STEP>'

ibmcloud iam oauth-tokens --output JSON | jq -r .iam\_token

export IAM\_TOKEN=$(ibmcloud iam oauth-tokens --output JSON | jq - r .iam\_token)

echo $IAM\_TOKEN

curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header "Authorization: $IAM\_TOKEN" -d '{"input\_data": [{"fields": ["sepal\_length", "sepal\_width", "petal\_length","petal\_width"],"values": [[5.1,3.5,1.4,0.2], [3.2,1.2,5.2,1.7]]}]}' $SCORING\_ENDPOINT

Step 6: Monitor your deployed model with IBM Watson OpenScale

IBM® Watson OpenScale tracks and measures outcomes from your AI models, and helps ensure they remain fair, explainable, and compliant wherever your models were built or are running. Watson OpenScale also detects and helps correct the drift in accuracy when an AI model is in production.

For ease of understanding, the tutorial concentrates only on improving the quality (accuracy) of the AI model through Watson OpenScale service.

Provision IBM Watson OpenScale service

In this section, you will create a Watson OpenScale service to monitor the health, performance, accuracy and quality metrics of your deployed machine learning model.

Create a IBM Watson OpenScale service

Select a region preferably Dallas. Create the service in the same region where you created the Machine Learning service.

Choose Lite plan.

Set the service name to watson-openscale-tutorial. Select a resource group.

Click Create.

Once the service is provisioned, Click Manage on the left pane and click Launch Application.

Click on Manual setup to manually setup the monitors. Selecting a deployment

In this section, as part of preparing your model for monitoring you will set up and enable monitors for each deployment that you are tracking with IBM Watson OpenScale.

By clicking on the Edit icon on the Database tile, choose Free lite plan database as your Database type and click Save. This is to store your model transactions and model evaluation results.

Click on Machine learning providers

Click on Add machine learning provider and click the edit icon on the Connection tile.

Select Watson Machine Learning(V2) as your service provider type. In the Deployment space dropdown, select the deployment space iris\_deployment\_space you created above.

Leave the Environment type to Pre-production. Click Save.

On the far left pane:

Click the icon for Insights dashboard(first icon) to add a deployment

Click on Add to dashboard to start the wizard on the Select model location page. On the Deployment spaces tab click on the iris\_deployment\_space radio button Click Next

On the Select deployed model page:

Click iris\_deployment Click Next

On the Provide model information page:

Data type: Numerical/categorical Algorithm type: Multi-class classification Click View summary

Click Finish

The iris\_deployment pre production dashboard is now displayed. Click Actions > Configure monitors

Click the pencil icon on the Training data tile to start the wizard. In the Select configuration method page

Click Use manual setup Click Next

In the Specify training data method page

For Training data option choose Database or cloud storage For Location choose Cloud Object Storage

For Resource instance ID and API key, run the below command in the Cloud Shell. Make sure to change the value after --instance-name to match the name of the Object Storage instance you have been using for this tutorial.

Copy and paste the Credentials resource\_instance\_id. It will begin with crn and end with two colons ::.

Copy and paste the Credentials api key without any trailing spaces. Click Connect.

Select the Bucket that starts with irisproject-donotdelete-. Select iris\_initial.csv from the Data set dropdown.

Click Next

In the Select the feature columns and label column method page

The defaults should be correct. Species as the Label/Target and the rest as Features.

Click Next

In the Select model output method page

The defaults should be correct, prediction for Prediction and probability for Probability.

Click View summary Click Finish

Click the pencil icon on the Model output details tile to start the wizard. In the Specify model output details method page

The defaults should be correct. Click Save

On the left pane, click on Quality under Evaluations and click the edit icon on the Quality thresholds tile

In the Quality thresholds page set the following values:

Accuracy 0.98 Click Next

In the Sample size page

Set Minimum sample size to 10 Click Save

On the left pane, Click on Go to model summary

The quality monitor (previously known as the accuracy monitor) reveals how well your model predicts outcomes.

As the tutorial uses a small dataset, configuring Fairness and Drift won't have any impact.

Evaluate the deployed model

In this section, you will evaluate the model by uploading a iris\_retrain.csv file which contains 10 instances of each species. Download iris\_retrain.csv.

Click on Actions and then Evaluate now.

Choose from CSV file as your import option and click on browse, upload the iris\_retrain.csv file.

Click and click on Upload and evaluate.

After the evaluation is completed, you should see the dashboard with different metrics.

To understand the quality metrics, refer to Quality metric overview

ibmcloud resource service-key $(ibmcloud resource service-keys --instance-name "cloud-object-storage-tutorial" | awk '/WDP-Project-Management/ {print $1}')

Step 7: Remove resources

Navigate to IBM Cloud® Resource List.

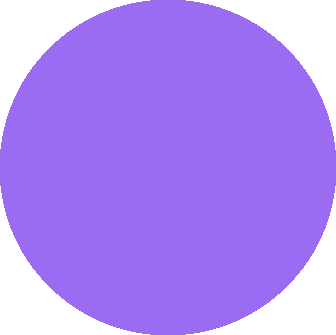
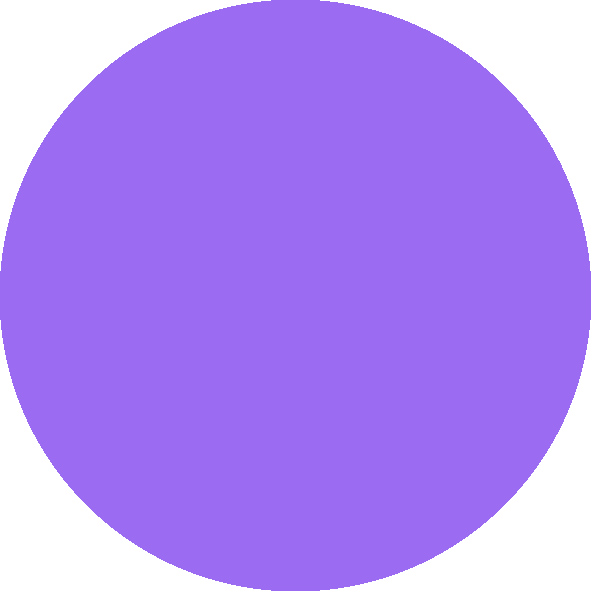
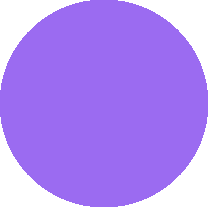
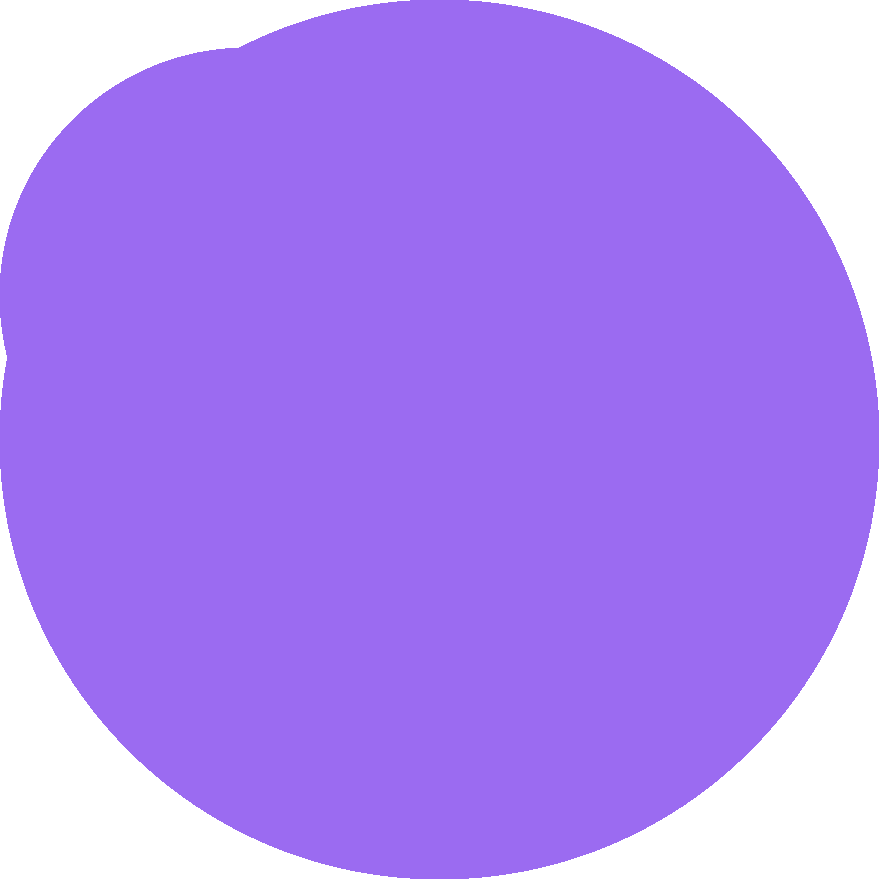
Under Name, enter tutorial in the search box.

Delete the services which you created for this tutorial.

Deploying Machine Learning Models in IBM Watson Studio Cloud as APIs



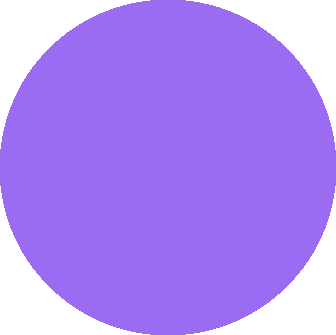
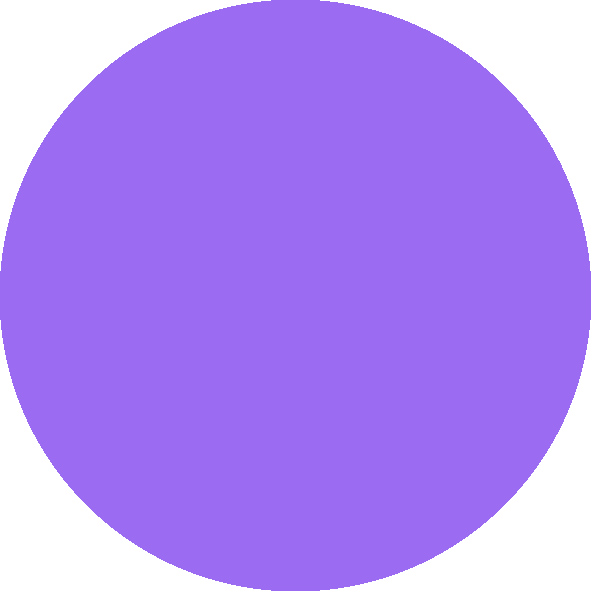
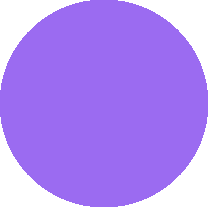
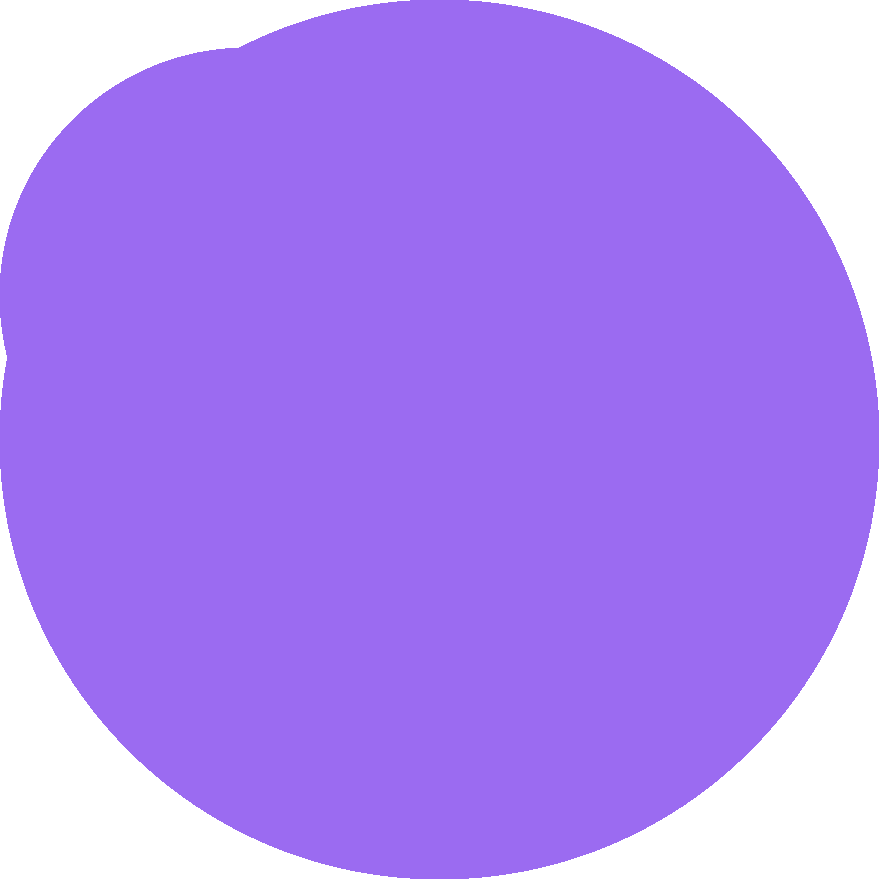
# Introduction



* Assuming that we've trained a good model, and want to deploy it, the immediate challenge is: how to represent the built model and deliver it for the deployment step. Here are a couple of options:

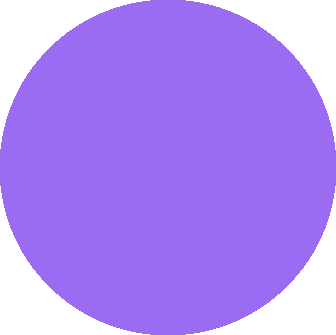
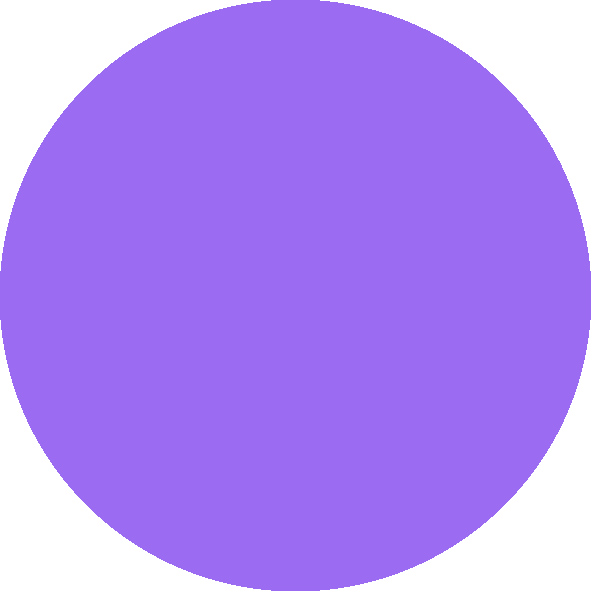
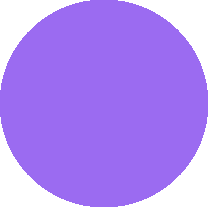
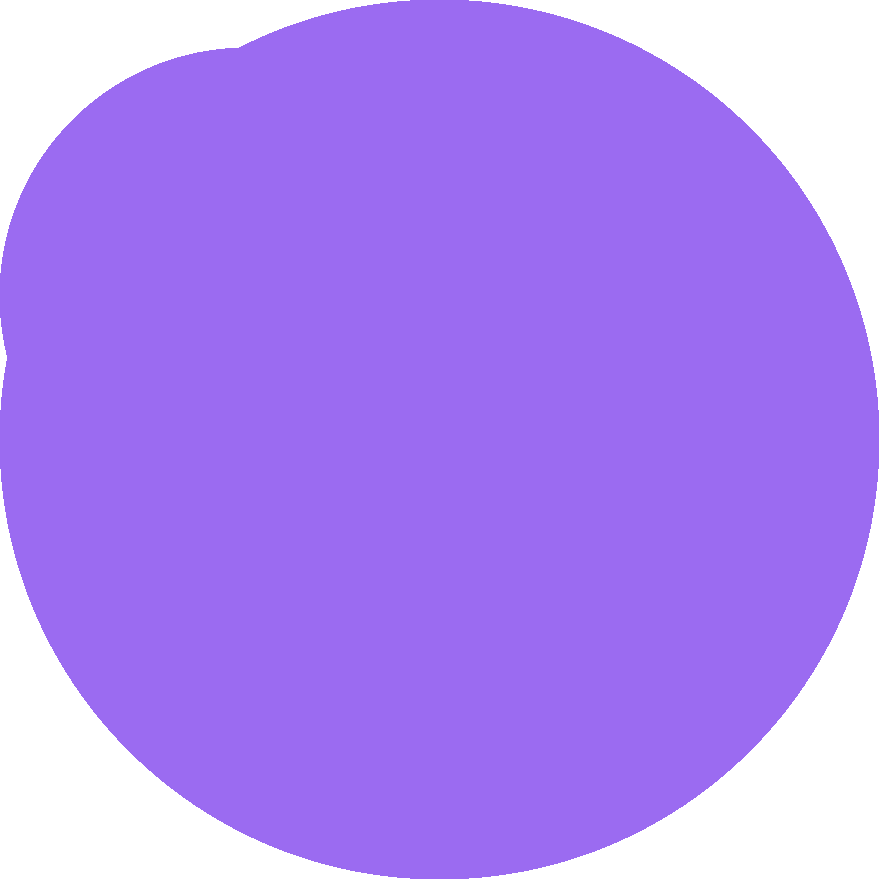
Model coefficients transfer approach, such as rewriting the scoring process with model coefficients in a "production language" like Java. This often causes the model outcomes to change and it takes a significant amount of time.

Model native serialization method, such as "pickle" of Python, and "save" function of Spark. It requires compatible environments on development and production like programming language, dependencies. It also leads to difficult configurations in production, and it might be slow and heavyweight.



* Predictive Model Markup Language (PMML) is the leading standard for statistical and data mining models and supported by over 20 vendors and organizations. PMML allows different statistical and data mining tools to speak the same language. With PMML, it is easy to develop a model on one system that uses one application and deploy the model on another system that uses another application.
* The models in PMML format provide flexibility to users, and in this blog, we discuss the deployment of models in PMML format.

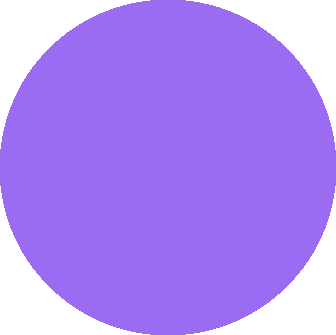
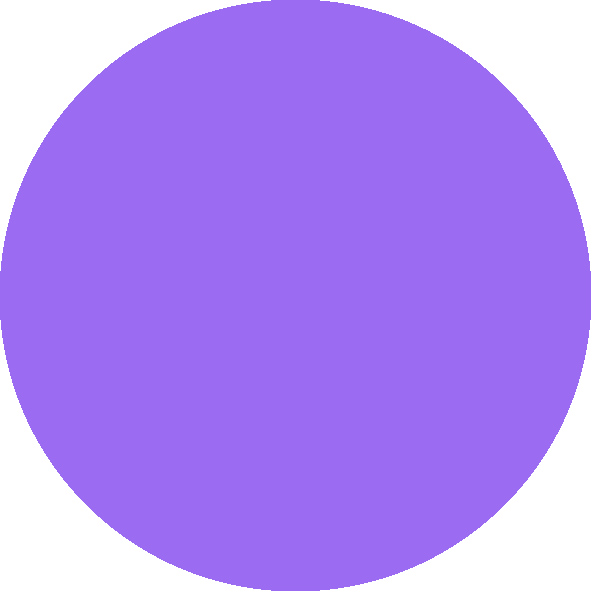
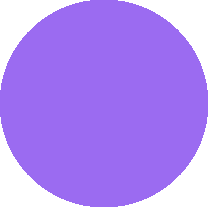
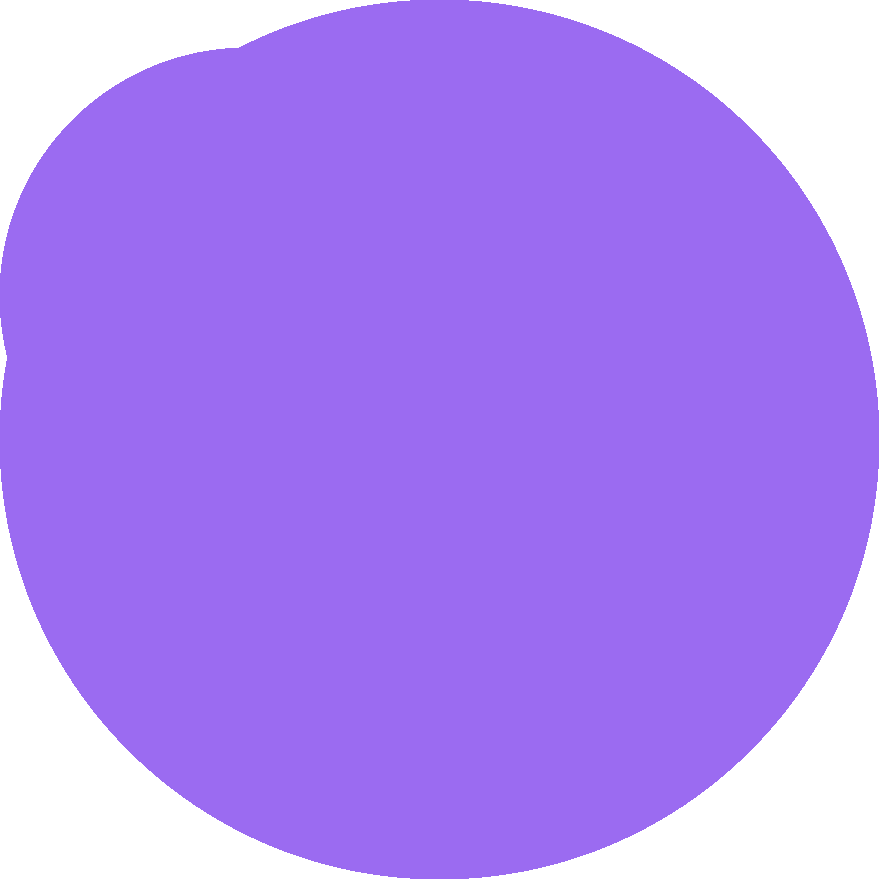
IBM Watson Studio is an integrated environment that is designed for AI and machine learning. It supports the whole process of machine learning and data mining practice: from data capture, data cleaning, data preparation, to feature engineering. Watson studio also does the model building via various machine learning technologies, for example Spark, Python and SPSS, and the model deployment as well as the model monitoring and retraining. When we say model deployment, Watson Studio integrates the SPSS Scoring Engine, which is a PMML-compliant engine. It is a solid pure Java library, and picks up many optimizations of computation, serves SPSS family more than ten years.



* + Association Rules
  + Cluster Models
  + General Regression
  + K-Nearest Neighbors
  + Mining Model
  + Naïve Bayes
  + Neural Network
  + Regression
  + Ruleset
  + Scorecard
  + Trees
  + Vector Machine

These models can have different versions and can be started via REST APIs. In this session, you will learn how to deploy PMML models in the IBM Watson Studio cloud and how to use them efﬁciently in applications.

# IBM Watson Studio



* Click the IBM Watson link in the header to navigate to the Watson Studio home panel.
* Click **New project**.
* Choose a project type "Standard":
  + If you want to train complex neural networks using experiments, choose a "Deep Learning" project
  + For all other machine learning work, choose the "Modeler" project type
* If you don't already have any of the required services, such as Watson Machine Learning and IBM Cloud Object Storage, new service instances are created.

- IBM **Watson Studio** Projects Tools Community Services Manage Support Docs 6

**CD**

## New project

**Define proj'ect details**

Na.me

ModelsDeployment

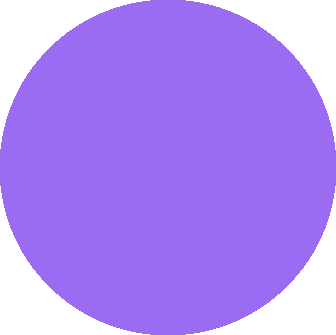
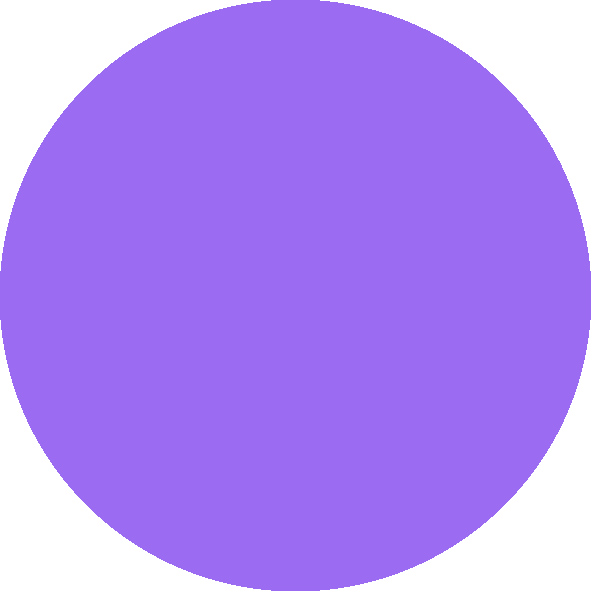
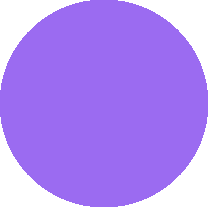
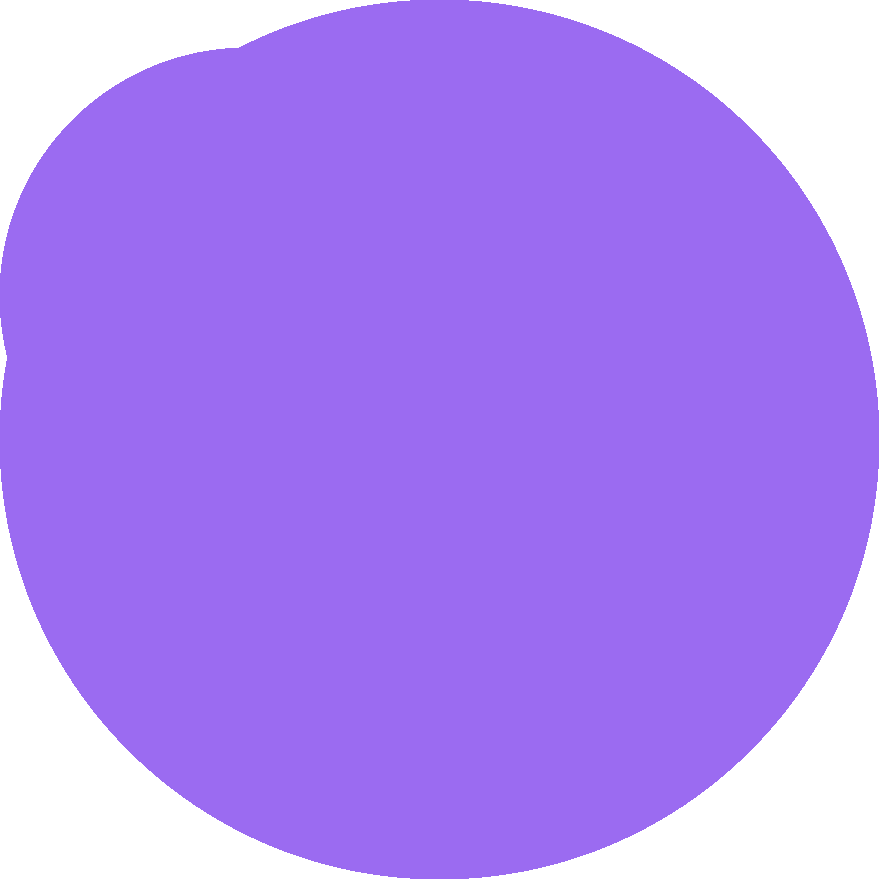
**84**

**Description**

This isa demo prnject for models deploymeint

**2956**

**Cance8l**



**To upload a PMML model**

1. From the Assets view of your project, click **New Watson Machine Learning model**.
2. Select **From ﬁle** as the model type.
3. Upload your PMML (.xml) ﬁle when you’re prompted. In the example, my PMML model is exported from SPSS Modeler. It is a CHAID model based on the known Iris Data Set. The model will be validated.

After PMML ﬁle is successfully validated, enter the required model name and optional descriptions.

1. Click **Create**. The model page will be open once it is imported successfully. The default `Overview` page shows general information about the model, including model type, label column, and a list of features.

IBM **Watson Studio** Projects Tools Community Services Manage Support Docs O b s's Account

##### **0** PMML filesuccessfully validated.

Define model details

Name

##### iriswclas

Select model type

Q Model builder **e** From file Q From sample

Model File

iriswclass.xml **O mJ**

##### 90

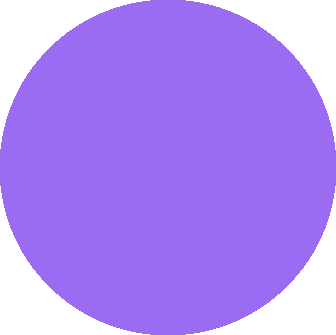
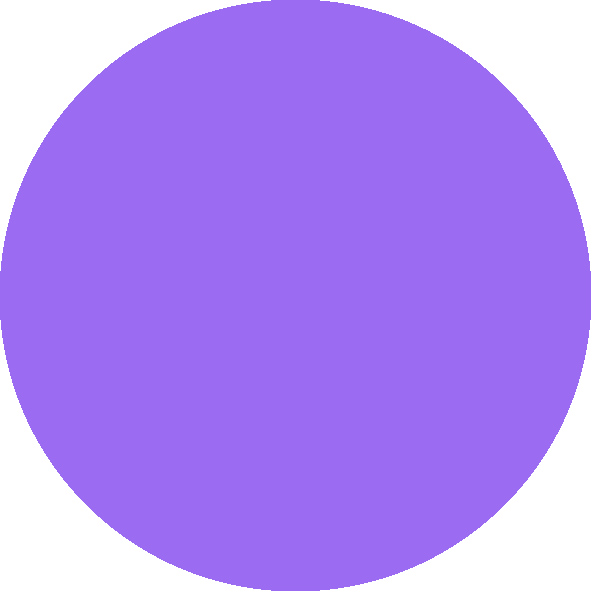
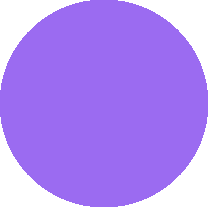
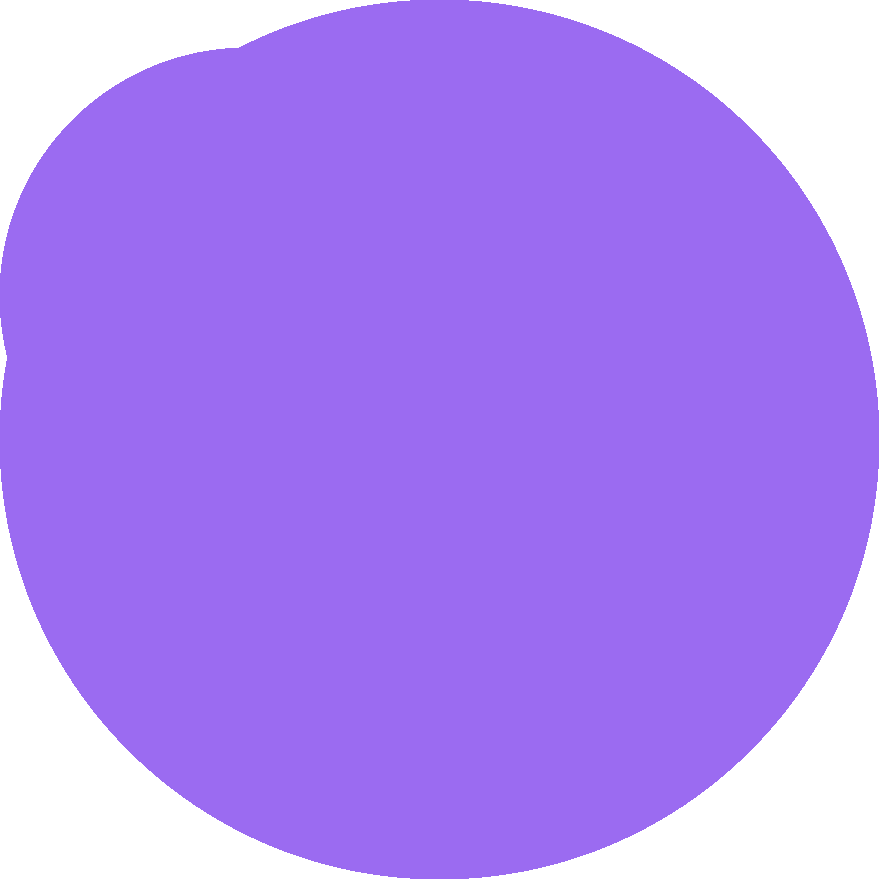
Description

Thisisa CHAID model

279

Cancel

Create



**To deploy the model**

* On the model page, click **Add Deployment**.
* Enter a deployment name, and optional descriptions.
* Select **Web service** deployment type and it exposes REST APIs for the deployed model. When the deployment phase is successful, we can open the deployment page to test the scoring function.

1Crea.te Deployment



NI m,



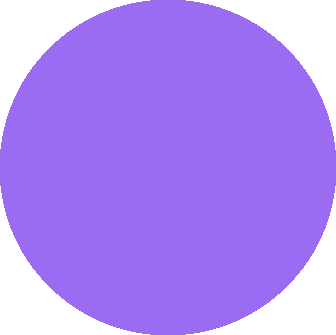
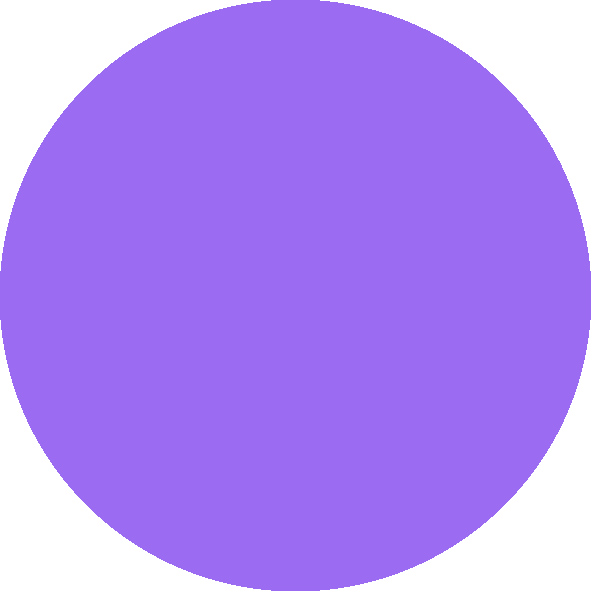
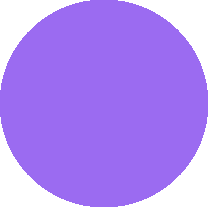
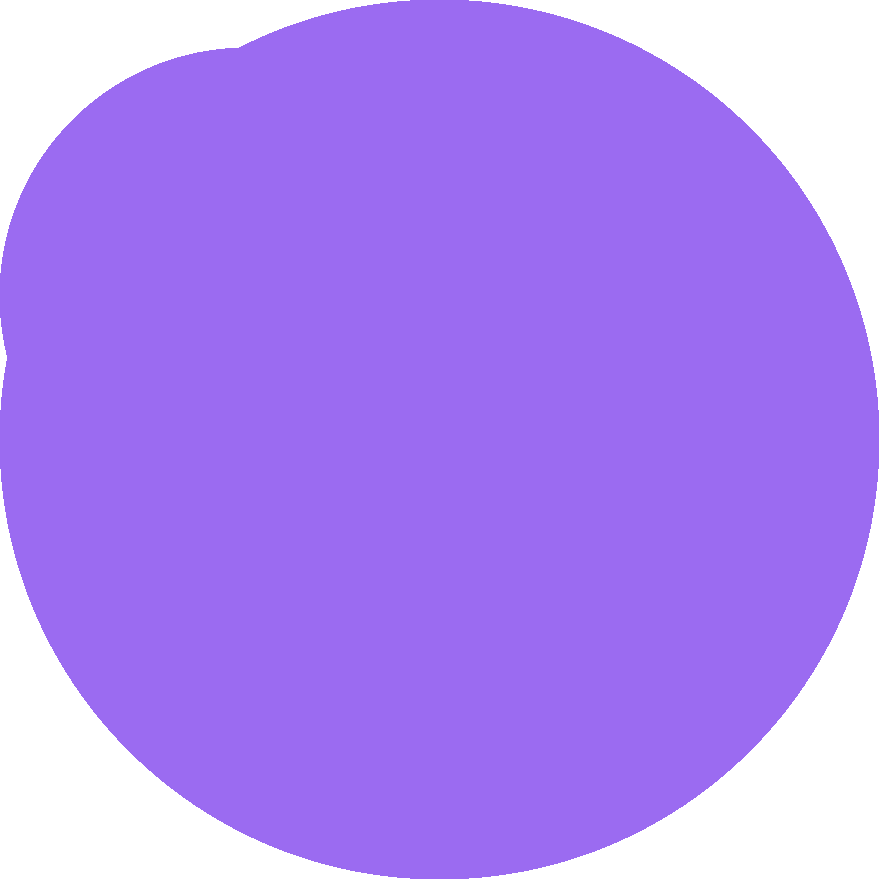
•.· sell'ipti1cm

D ploym nt 'typ

**e** Wob . rvic

nc:: l **5 v**

1. Click **Te,st**
2. Entler 1data ,and click **Predict** to mlake a predictio1n.



**To deploy the model**

* + On the model page, click **Add Deployment**.
  + Enter a deployment name, and optional descriptions.
  + Select **Web service** deployment type and it exposes REST APIs for the deployed model. When the deployment phase is successful, we can open the deployment page to test the scoring function.

lmp,mi en.a. ion T



"Id .





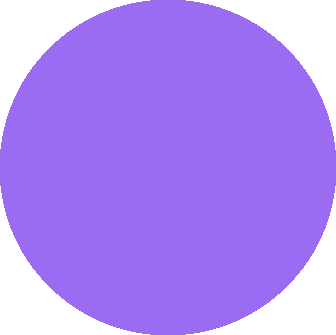
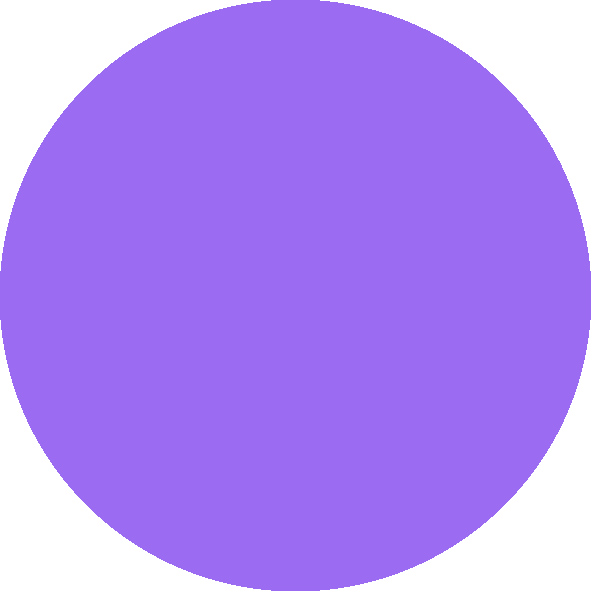
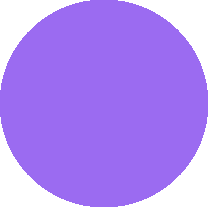
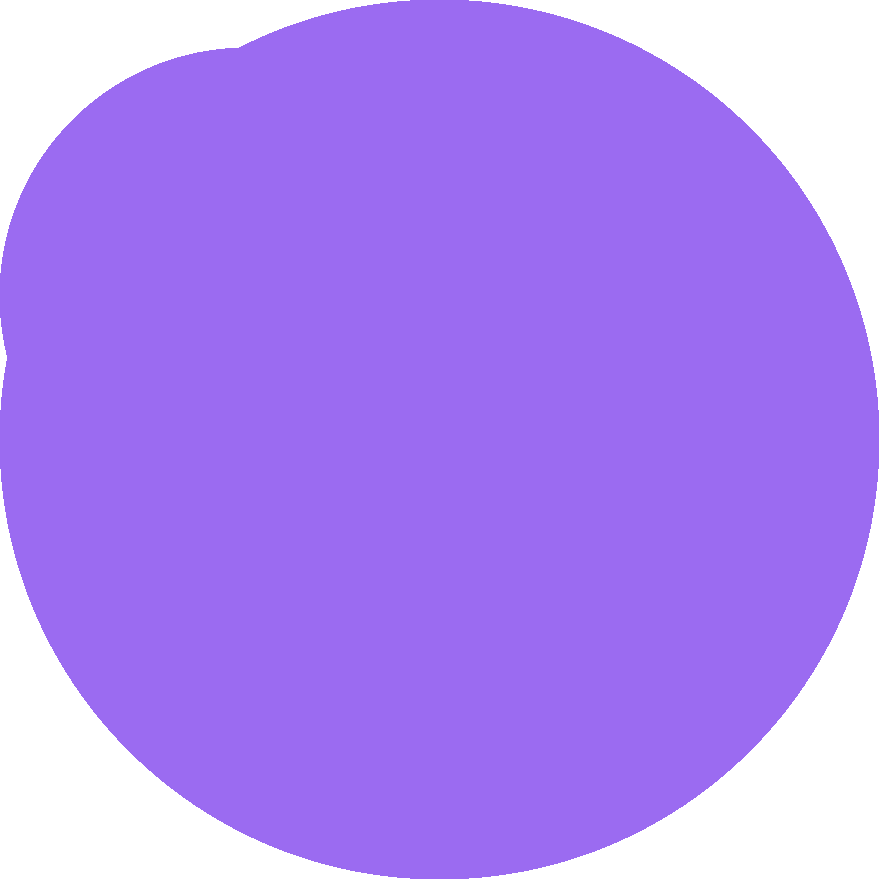
2.  ,.,



]

•



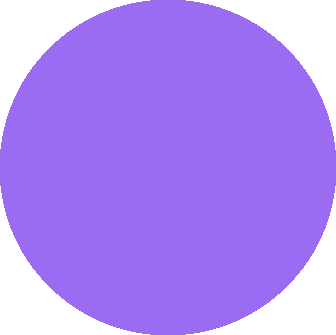
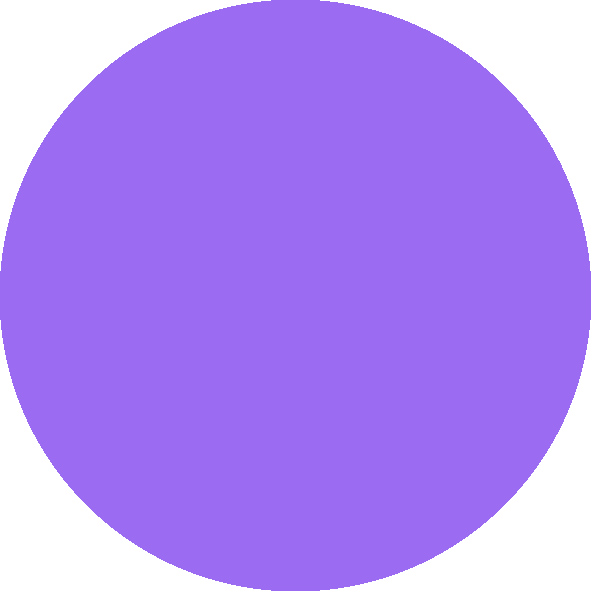
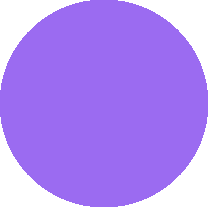
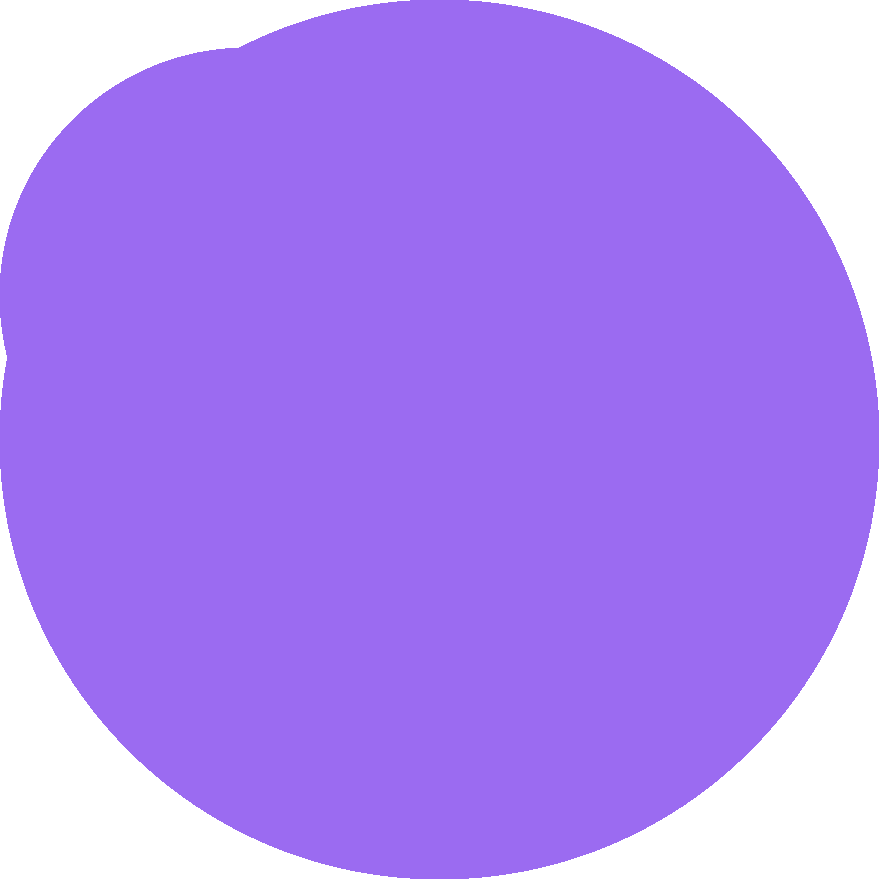


* + Since the web service deployment exposes standard REST APIs, we can connect to any client.

1. Click **Implementation** to access information, such as endpoint and authorization of service. The Implementation page displays all of the information you can use and several code snippets of different languages for your reference.

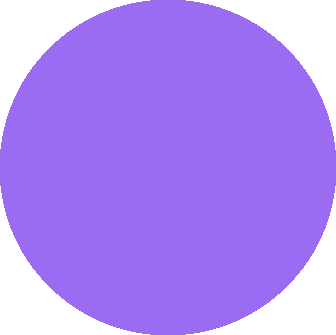
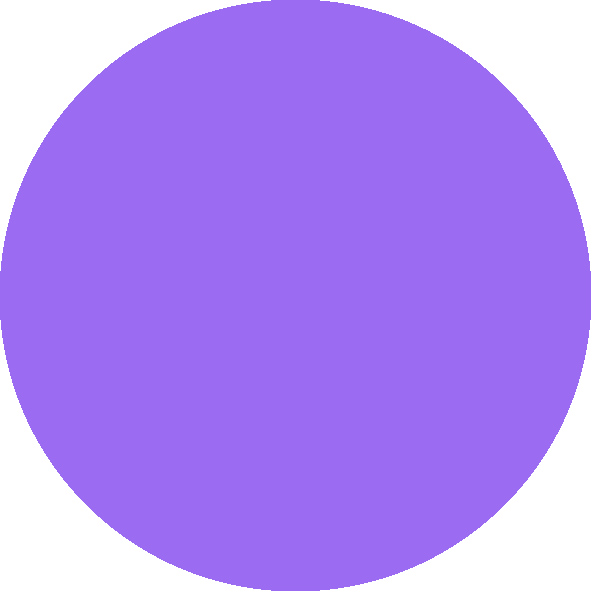
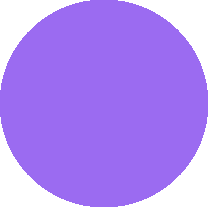
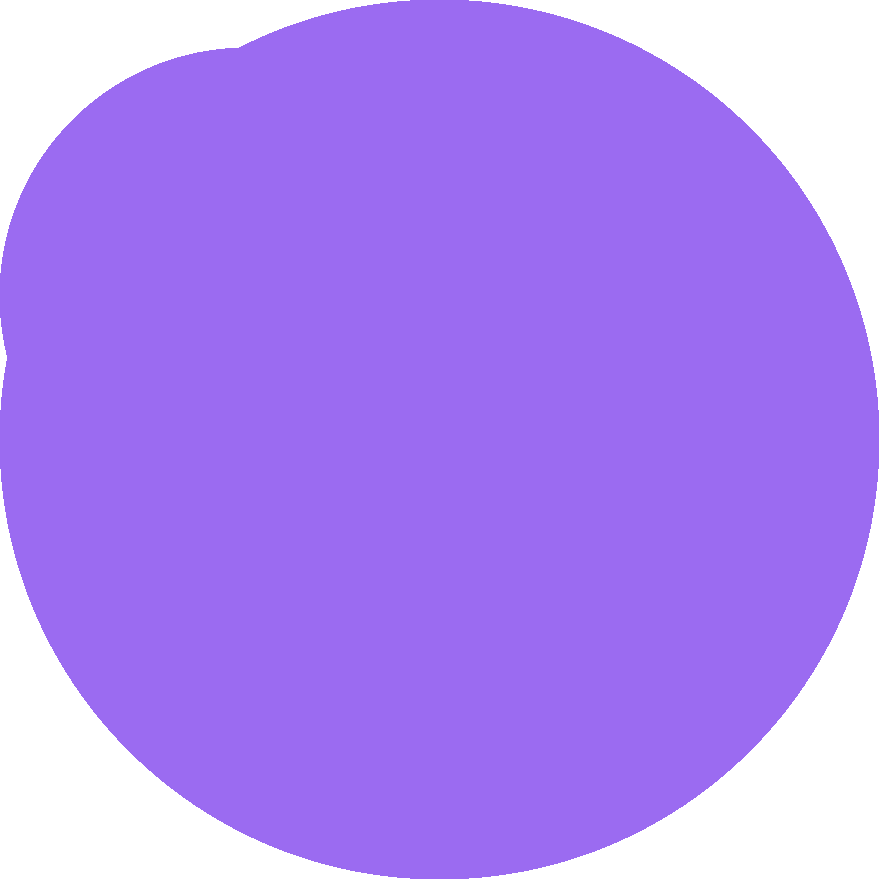
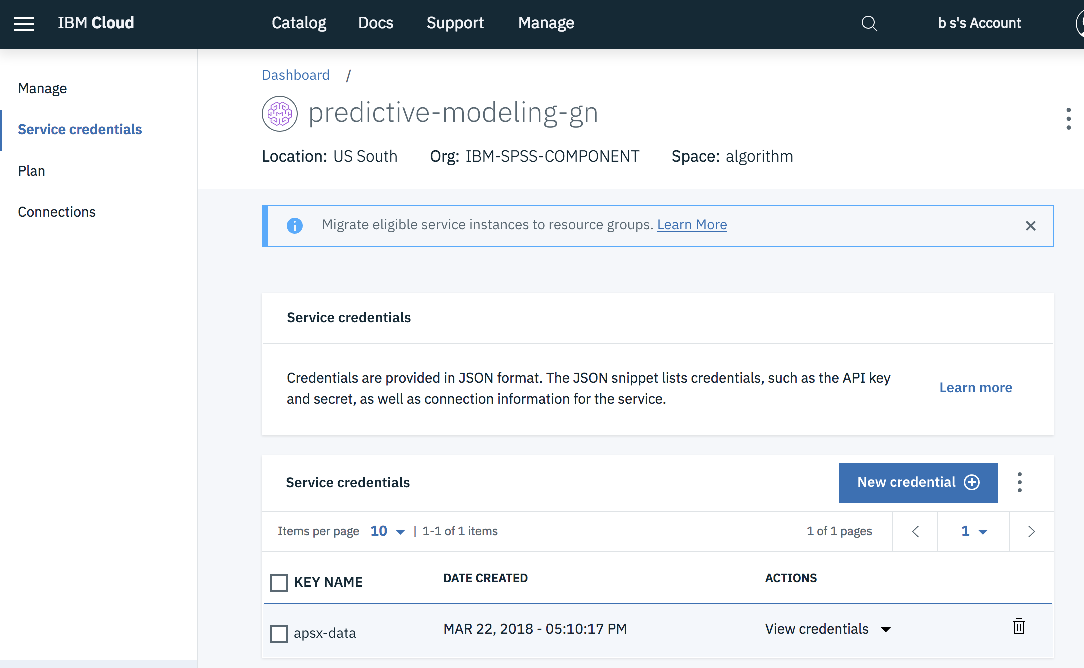
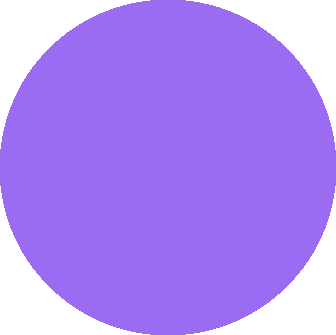
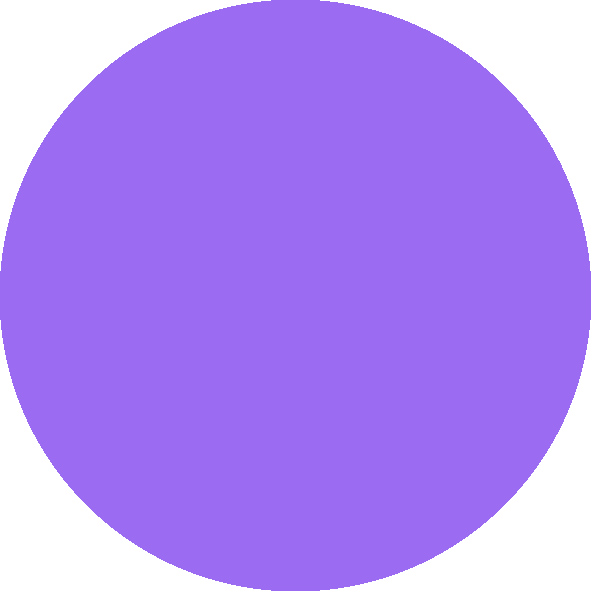
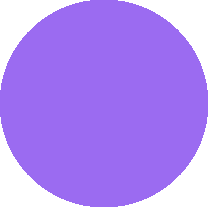
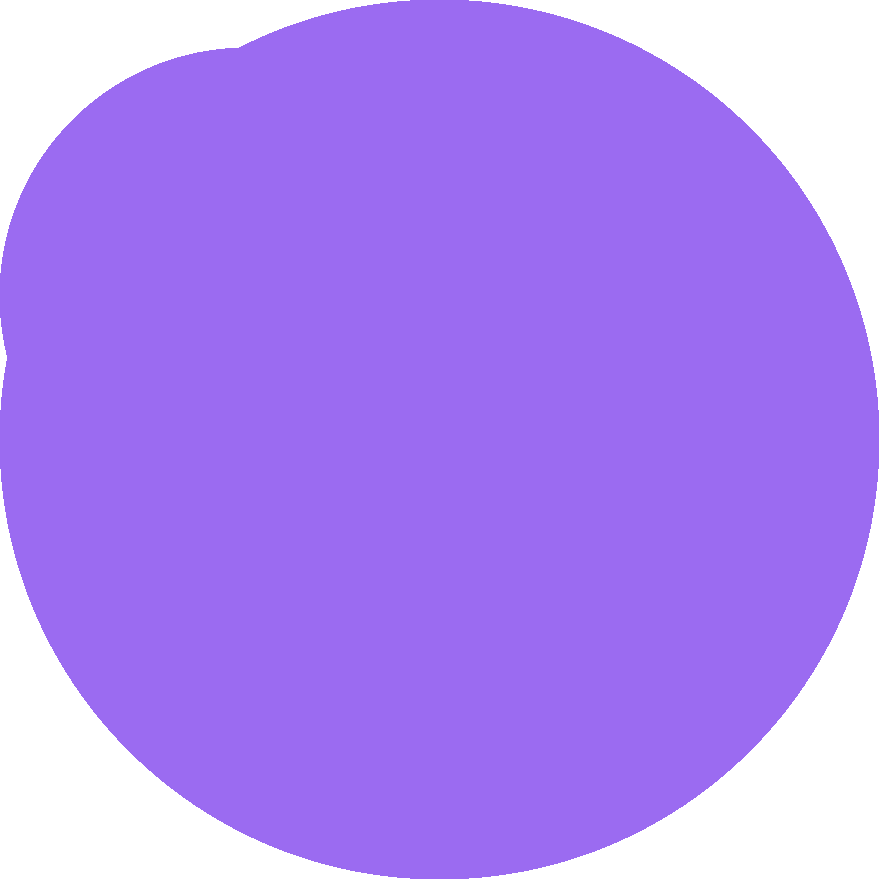
Now let’s look at an example of using curl. Based on the descriptions, you need to retrieve three variables from the service credentials associated with your IBM Cloud Watson Machine Learning Service instance.

1. Click **Overview** of deployment to get the name of the associated machine learning service, and then go to its home page.



* + Click **New credential** if there are no existing credentials, click **View credentials** to get the url, username, and password variables.

1. Start a shell to export the url, user name, and password variables.
2. Verify the variables to get a valid token and initiate a request with a record payload to make the prediction.



* + . Start a shell to export the url, user name, and password variables.

10. Verify the variables to get a valid token and initiate a request with a record payload to make the prediction.

I I 0 bsong- -bash-156x32

bsongs-mbp: bsong$ export WML\_SERVICE\_CREDENTIALS\_USERNAME=a871380c-7647-42e2-8949-4147afedd661 bsongs-mbp: bsong$ export WML\_SERVICE\_CREDENTIALS\_PASSWORD=7012f3ae-86dd-46a5-a85a-le4d82cff8c5 bsongs-mbp: bsong$ export WML\_SERVICE\_CREDENTIALS\_URL=https://ibm-watson-ml.mybluemix.net bsongs-mbp: bsong$

bsongs-mbp: bsong$ curl --basic --user $WML\_SERVICE\_CREDENTIALS\_USERNAME:$WML\_SERVICE\_CREDENTIALS\_PASSWORD $WML\_SERVICE\_CREDENTIALS\_URL/v3/identity/token

{11token11:11eyJhbGciOiJSUzUxMiisinR5cCI6IkpXVCJ9,eyJ0ZW5hbnRJZCI6IjRmNTZlNDglLTgxNWEtNDZiZC04Y2JkLTkzYTc5NjRhMjdh05Isimluc3RhbmNlSWQiOiI0ZjU2ZTQ4N504MTVhLTQ2

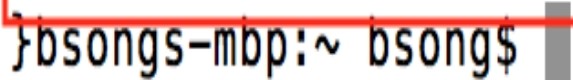
mQtOGNiZC05M2E30TY0YTI3YTkiLCJwbGFuSWQiOiizZjZhY2Y0MyllZGU4LTQxM2EtYWM2051mOGFmM2JiMGNiZmUiLCJyZWdpb24iOiJlcylzb3V0aCisinVzZXJJZCI6ImE4NzEzODBjLTc2NDctNDJl i040TQ5LTQxNDdhZmVkZDY2MSisimlzcyI6Imh0dHBzOi8vdXMtc291dGgubWwuY2xvdWQuaWJtlmNvb592My9pZGVudGl0e5IsimlhdCI6MTU0MDk3NjkwNSwiZXhwijoxNTQxMDA1NzAlfQ,lB4b0EZmlp V8tVlRMqDHaA1M71nrXF-kAshtJbHuuf95RiEOiu-vGFur0-4dk73Kncx8r8PQG\_0NIOOkILDI0tKMtWrx2L7C3QiyN70Tvj-CK9vORDxZ5AJkQs0jdlVom58lBrAYPdGWivih7w6oFhAYmnrFhWGCbGatD Uab8yDd-ITo3JNxIIjwapG5vpz5jbII5nx-xmFLY5G594G2UairrF89aaAEEVtNyIQpOv-6lorDx2r\_BsPEDQZRVRcz0e2M9PkblFnOoichFa0t\_sDskz3No3F01Ky7VPVBGi5q272RewaHeAt8WgosE8fD L9Elpa38DT2xh6nwIJjA"}bsongs-mbp: bsong$

bsongs-mbp: bsong$

bsongs-mbp: bsong$ export WML\_AUTH\_TOKEN="eyJhbGciOiJSUzUxMiisinR5cCI6IkpXVCJ9.eyJ0ZW5hbnRJZCI6IjRmNTZlNDglLTgxNWEtNDZiZC04Y2JkLTkzYTc5NjRhMjdh05Isimluc3R bmNlSWQiOiI0ZjU2ZTQ4N504MTVhLTQ2YmQtOGNiZC05M2E30TY0YTI3YTkiLCJwbGFuSWQiOiizZjZhY2Y0MyllZGU4LTQxM2EtYWM2051mOGFmM2JiMGNiZmUiLCJyZWdpb24iOiJlcylzb3V0aCisinV ZXJJZCI6ImE4NzEzODBjLTc2NDctNDJlMi040TQ5LTQxNDdhZmVkZDY2MSisimlzcyI6Imh0dHBzOi8vdXMtc291dGgubWwuY2xvdWQuaWJtlmNvb592My9pZGVudGl0e5IsimlhdCI6MTU0MDk3NjkwNSw ZXhwijoxNTQxMDAlNzAlfQ,lB4b0EZmlpV8tVlRMqDHaA1M71nrXF-kAshtJbHuuf95RiEOiu-vGFur0-4dk73Kncx8r8PQG\_0NIOOkILDI0tKMtWrx2L7C3QiyN70Tvj-CK9vORDxZ5AJkQs0jdlVom58l rAYPdGWivih7w6oFhAYmnrFhWGCbGatDFUab8yDd-ITo3JNxIIjwapG5vpz5jbII5nx-xmFLY5G594G2UairrF89aaAEEVtNyIQpOv-6lorDx2r\_BsPEDQZRVRcz0e2M9PkblFnOoichFa0t\_sDskz3No3F 1Ky7VPVBGi5q272RewaHeAt8WgosE8fDLL9Elpa38DT2xh6nwIJjA"

bsongs-mbp: bsong$

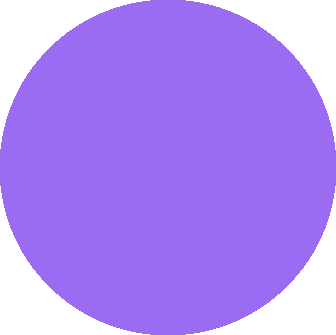
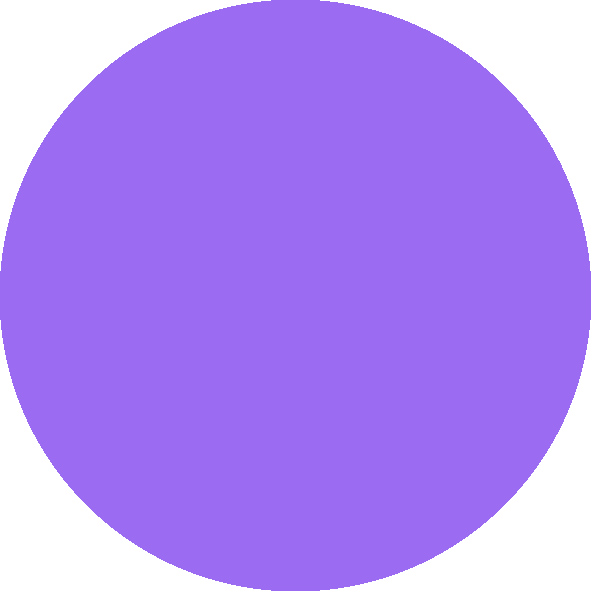
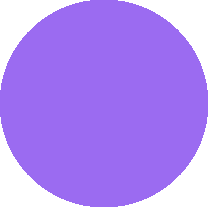
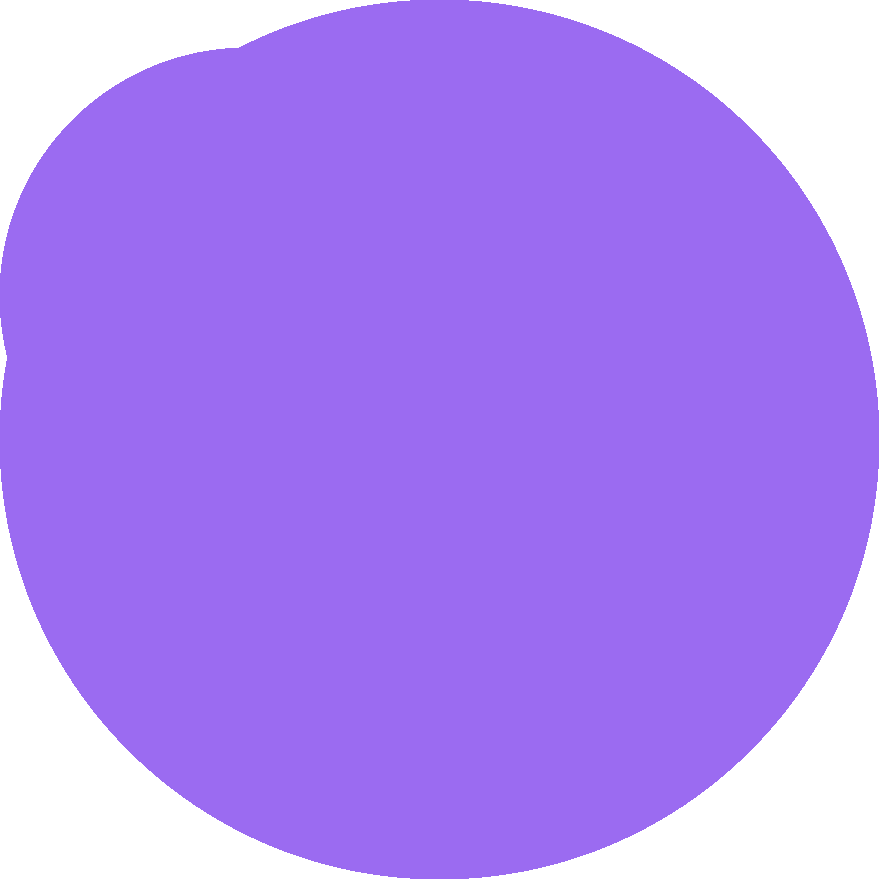
bsongs-mbp: bsong$ curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' --header "Authorization: Bearer $WML\_AUTH\_TO KEN" -d '{"fields": ["petal\_width", "sepal\_width", 11petal\_length11],11values11: [[1.2, 2,5, 5,2]]}' https://ibm-watson-ml.mybluemix.net/v3/wml\_instances/4f56e 85-815a-46bd-8cbd-93a7964a27a9/deployments/b68542a6-7220-434b-85bb-d30b48ba15e2/online



"fields": [[11$R-class11, "$RC-class", "$RP-class", 11$RP-Iris-setosa11, 11$RP-Iris-versicolor11, 11$RP-Iris-virginica11, "$RI-class"]],

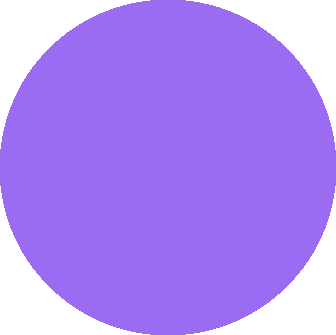
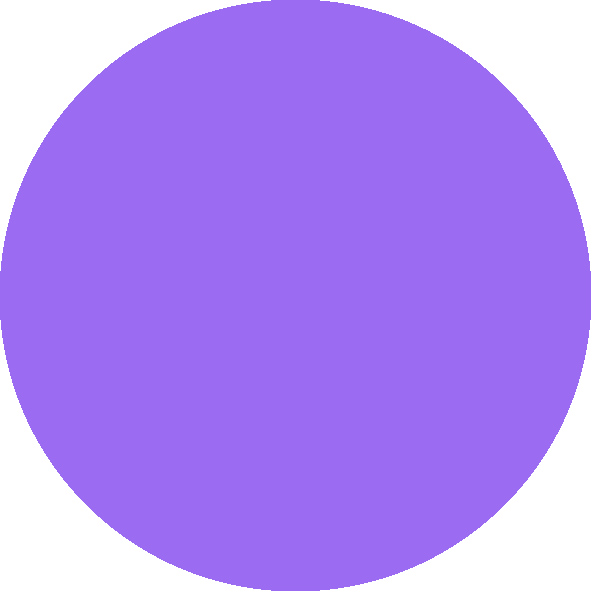
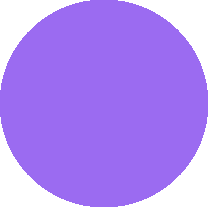
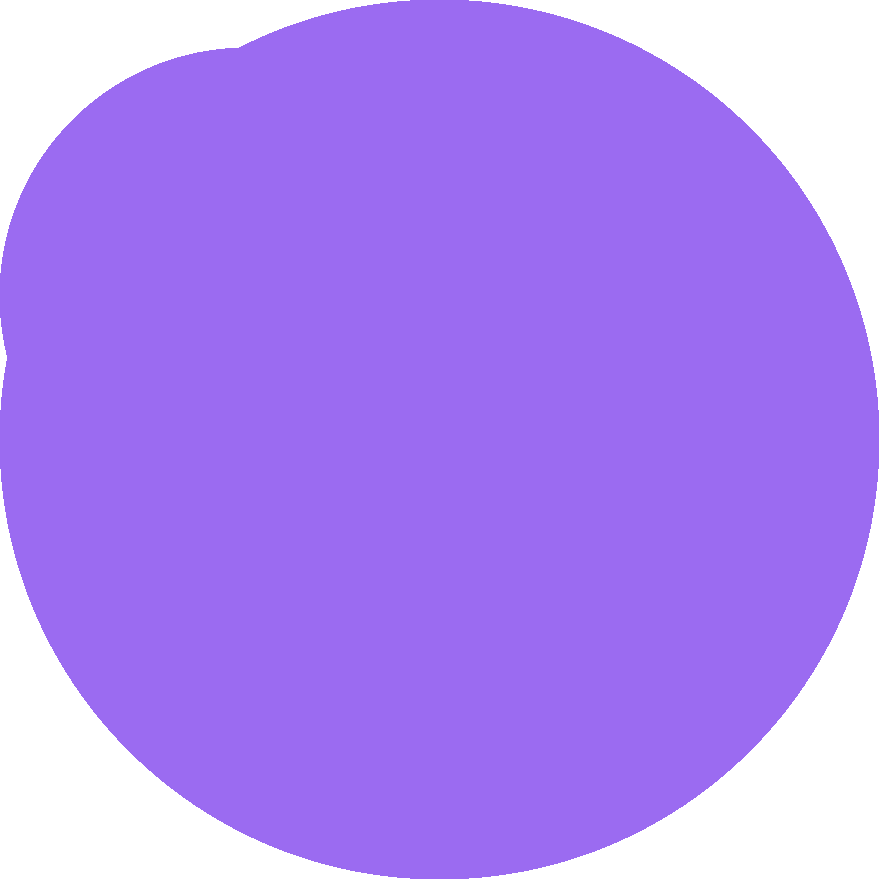
"values": [["Iris-versicolor", 1.0, 1.0, 0,0, 1.0, 0,0,

11511]]



# As you can see, the results are same as ones produced by the test function. At this point, we hope you see how easy it is to deploy PMML Model into a REST API with Watson Studio.

DONE BY



**Name: HARIHARAN P**

**Year:3rd year Dept:CSE**

**College: University College Engineering Thirukkuvalai**