# Hugging Face Hub end-to-end Tutorial

- *Sections 2 to 4 are documented in the Jupyter Notebook* `1_iris_classification.ipynb`
- *Section 5 in* `2_upload_dataset.ipynb`
- *Section 6 in* `3_upload_model.ipynb`
- *Section 7 in script* `4_gradio.ipynb`

The notebook `1_iris_classification.ipynb` trains a Decision Tree classifier. The tutorial contains the modified code for the Logistic Regression model

- **YOUR TASKS for accomplishing this tutorial** are:
    1. Adapt the code in the notebook for this other classifier (**section 4**)
    2. Add the new Logistic Regression model to the same Hugging Face model repository, giving the option to the user of using one or the other (**section 6**)

**NOTE**: *The code snippets in this tutorial are minimal and orientative. Refer to Jupyter Notebook* `1_iris_classification.ipynb` *for the complete code.*

## 1. Project Setup

1. **Create/Activate a Python Environment**

    - Install required packages

    ```
    pip install scikit-learn huggingface_hub datasets gradio joblib
    ```

2. **Authenticate with Hugging Face**

    - Create an access token on Hugging Face
        - *Settings* > *Access Tokens*: Create new token, Write type
    - Use `huggingface-cli login` and paste your token
    - Alternatively, login from a Python script:

    ```
      import huggingface_hub
      huggingface_hub.login(token=...)`
    ```

---

## 2. Loading the Iris Dataset

You have Iris dataset available in path `./iris-HF/Iris.csv`, so you do not need to obtain it externally. For your reference, here are two ways to get the dataset using a Hugging Face repository ot the `datasets` library (equivalent to importing from Sklearn).

1. Clone the dataset from Hugging Face hub:

```
git lfs install
git clone https://huggingface.co/datasets/scikit-learn/iris
```

2. If you are inside Python you can alternatively use `datasets` HF library:

   ```
   from datasets import load_dataset
   ```

   Or load data from Scikit-Learn's built-in dataset:

   ```
   from sklearn.datasets import load_iris
   data = load_iris()
   ```

Afterwards inspect the Data:

- Print feature names, shapes, target distribution, etc.

---

## 3. Preparing and Splitting the Data

1. **Train/Test Split**
   - Use `train_test_split` from `sklearn.model_selection`.
   - Example:

     ```
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(
         data['data'], data['target'], test_size=0.2, random_state=42
     )
     ```

2. **Preprocessing**
   - Scale features or apply other transformations as needed.

---

## 4. Training a Logistic Regression Model

1. **Train the Model**

   ```
   from sklearn.linear_model import LogisticRegression

   model = LogisticRegression(max_iter=200)
   model.fit(X_train, y_train)
   ```

2. **Evaluate the Model**

- Generate predictions, compute accuracy:

  ```
  from sklearn.metrics import accuracy_score

  y_pred = model.predict(X_test)
  print("Accuracy:", accuracy_score(y_test, y_pred))
  ```

- Confusion matrix, classification report, etc.

---

## 5. Uploading the Dataset to Hugging Face

Refer to script `2_upload_dataset.ipynb`

1. **Create a Dataset Repository**

   - Using `huggingface_hub` to initialize a repo:

     ```
     from huggingface_hub import HfApi

     api = HfApi()
     repo_id = "username/iris-dataset"
     api.create_repo(repo_id=repo_id, repo_type="dataset")
     ```

2. **Push the Data Files**

   - Store the split data in CSV or Parquet.
   - Push the CSV file of the dataset:

   ```
    api.upload_file(
    path_or_fileobj="iris-HF/iris.csv",        # Local path to your CSV
    path_in_repo="iris.csv",           # File name/path in the repository
    repo_id= repo_id,     # The dataset repository on HF
    repo_type="dataset",               # repo_type="dataset" for a dataset repo
    # token="hf_your_token_here"         # Your personal HF token (not needed,
   because you're already logged in)
   )
   ```

   Or using the command line:

   ```
   huggingface-cli upload brjapon/iris . --repo-type=dataset
   ```

3. **Dataset Card**

   - Clone the repository locally:

```
   git clone https://huggingface.co/datasets/USERNAME/iris
```

- Create a `README.md` describing the dataset.
- Push it from the command line using git

```
git add README.md
git commit -m "Added README file"
git push
```

---

## 6. Saving and Uploading the Model to Hugging Face

Refer to script `3_upload_model.ipynb`

1. **Serialize Your Model**

```
import joblib
joblib.dump(model, "iris_logreg_model.joblib")
```

2. **Create a Model Repository**

```
from huggingface_hub import HfApi, HfFolder, upload_file

api = HfApi()

repo_id = "username/iris-logistic-regression"
api.create_repo(repo_id=repo_id, repo_type="model")
```

3. **Push the Model**

```
upload_file(
    path_or_fileobj="models/iris_logreg_model.joblib",
    path_in_repo="iris_logreg_model.joblib",
    repo_id=repo_id,
    repo_type="model"
)
```

4. **Model Card**
   - Clone the repository locally:

```
   git clone https://huggingface.co/datasets/USERNAME/iris-dt
```

- Create a `README.md` describing the dataset.
- Push it from the command line using git

```
git add README.md
git commit -m "Added README file"
git push
```

---

## 7. Creating a Hugging Face Space for Deployment

Refer to script `4_gradio.py`

1. **Choose a Framework (Gradio/Streamlit)**

   - We will use Gradio, that is more focused to Machine Learning apps. This is the version where we start from (that launches the Gradio app pointing to local file of the model):

   ```
   import gradio as gr
   import joblib
   import numpy as np

   model = joblib.load("iris_dt.joblib")

   def predict_iris(sepal_length, sepal_width, petal_length, petal_width):
       prediction = model.predict([[sepal_length, sepal_width,
   petal_length, petal_width]])
       return prediction[0]

   interface = gr.Interface(
       fn=predict_iris,
       inputs=["number", "number", "number", "number"],
       outputs="text"
   )

   if __name__ == "__main__":
       interface.launch()
   ```

Replace Gradio script with this version pointing to the model hosted in Hugging Face:

```
import gradio as gr
import joblib
import numpy as np
from huggingface_hub import hf_hub_download

HF_TOKEN = 'hf_your_token_here'  # Replace with your actual Hugging Face
token
```

```
model_path = hf_hub_download(
 repo_id="brjapon/iris-dt",
 filename="iris_dt.joblib",      # The model file stored in the HF repo
 repo_type="model"               # Could also be 'dataset' if you're storing
it that way
)

# Load the trained model
pipeline = joblib.load(model_path)

# Define a function that takes the four iris measurements as input
# and returns the predicted iris species label.
def predict_iris(sepal_length, sepal_width, petal_length, petal_width):
   # Convert the input parameters into a 2D list/array because
   # scikit-learn's predict() expects a 2D array of shape (n_samples,
n_features)
   input = np.array([[sepal_length, sepal_width, petal_length,
petal_width]])
   prediction = pipeline.predict(input)

   # Convert the prediction to the string label
   if prediction == 0:
      return 'iris-setosa'
   elif prediction == 1:
      return 'Iris-versicolor'
   elif prediction == 2:
      return 'Iris-virginica'
   else:
      return "Invalid prediction"

   interface = gr.Interface(
 fn=predict_iris,
 inputs=["number", "number", "number", "number"],
 outputs="text",
 live=True,
 title="Iris Species Identifier",
 description="Enter the four measurements to predict the Iris species."
)
```

2. **Deploying to Spaces**

   - Create a new Space on Hugging Face (click “New Space,” choose Gradio)
   - Clone the repository:

```
# When prompted for a password, use an access token with write permissions
git clone https://huggingface.co/spaces/brjapon/iris-space
```

   - Add the Python script of your Gradio app, and the Python dependencies, i.e. file `requirements.txt`:

```
git add app.py requirements.txt
git commit -m 'Add application file'
git push
```

  - Push your code (e.g. `app.py`, `requirements.txt`) directly from your local environment

```
git push
```

  - Wait for the build to finish, then test your live app

---

## 8. Testing and Sharing Your Deployed App

1. **Test the UI**
   - Manually input values.
   - Verify correctness of predictions.
2. **Share the Space URL**
   - A link for sharing the app can be obtained from the `Share via Link` button that you will see in the live app:
     - https://brjapon-iris-space.hf.space/?__theme=system&deep_link=TFBJ-sNX6lk
   - Collaborators and other users can directly interact with or fork your Space.