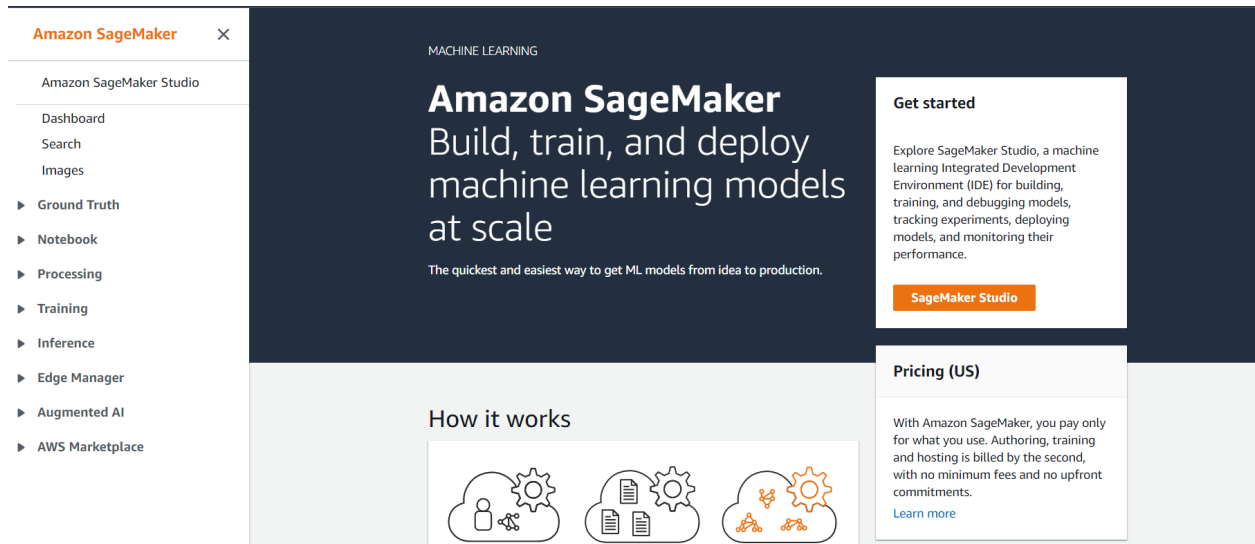


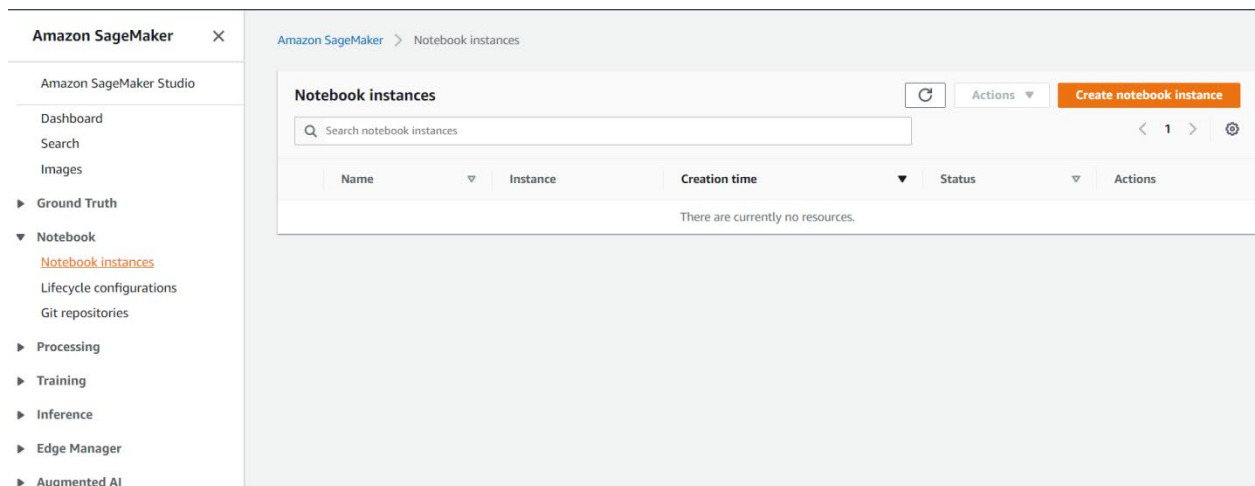
AWS SageMaker

Build, Train, and Deploy Machine Learning models at scale

1. Creating and Importing Data



AWS SageMaker Dashboard



NoteBook Instances Page

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

Mynotebook

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.m4.xlarge

Elastic Inference [Learn more](#)

none

Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). [Learn more](#)

Platform identifier [Learn more](#)

notebook-ml-v1

▶ Additional configuration

Creating Notebook Instance

Amazon SageMaker

Amazon SageMaker Studio

Dashboard

Search

Images

▶ Ground Truth

▶ Notebook

▶ Processing

▶ Training

▶ Inference

▶ Edge Manager

▶ Augmented AI

▶ AWS Marketplace

Amazon SageMaker > Notebook instances

Notebook instances

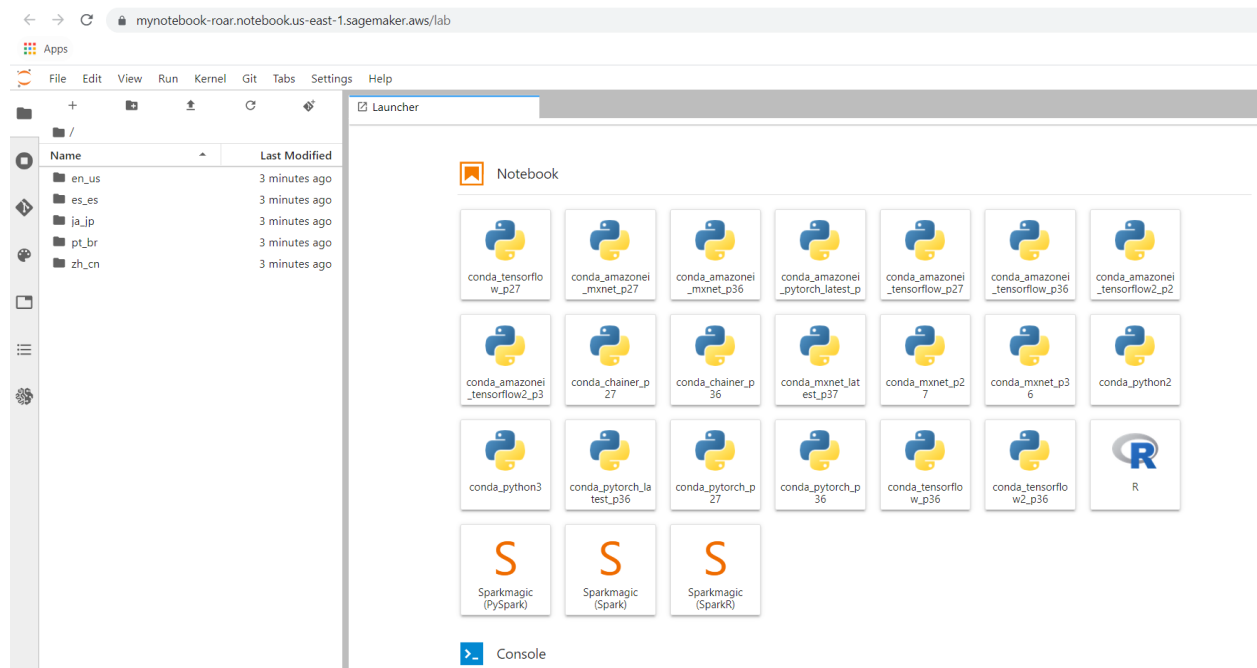
Search notebook instances

< 1 >

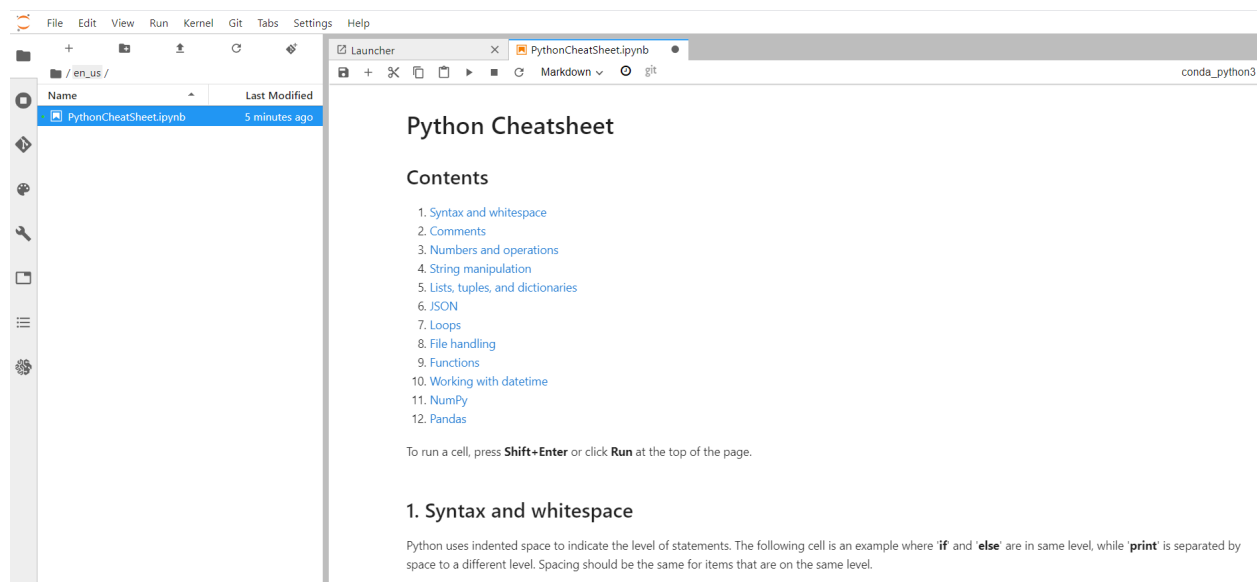
Create notebook instance

| Name | Instance | Creation time | Status | Actions |
|------------|--------------|------------------------|-----------|--|
| Mynotebook | ml.m4.xlarge | Sep 22, 2021 16:28 UTC | InService | Open Jupyter Open JupyterLab |

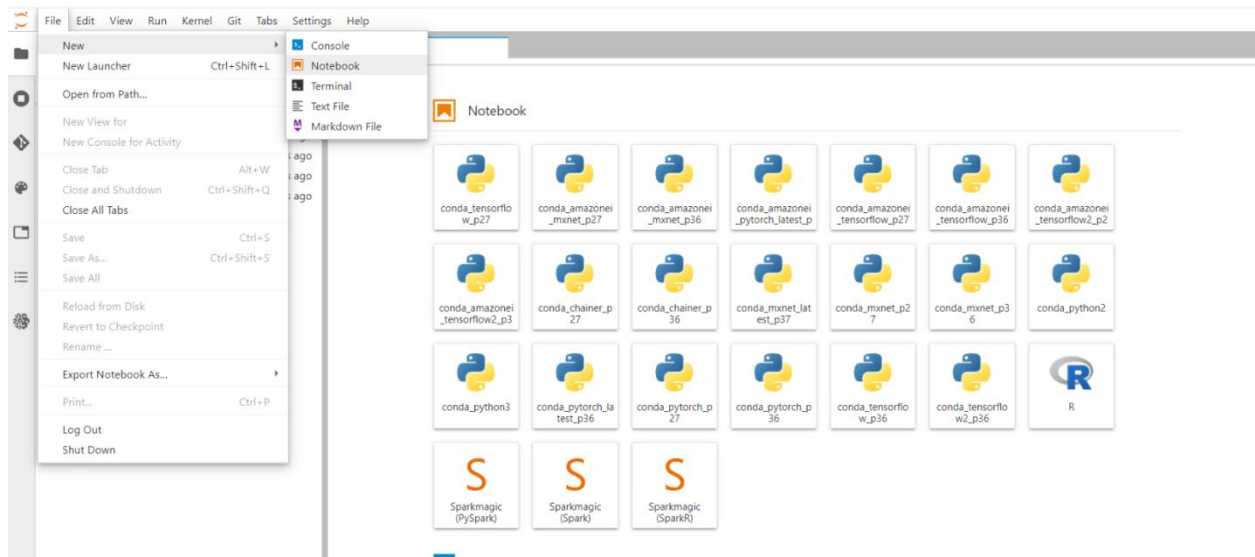
Notebook Successfully Created



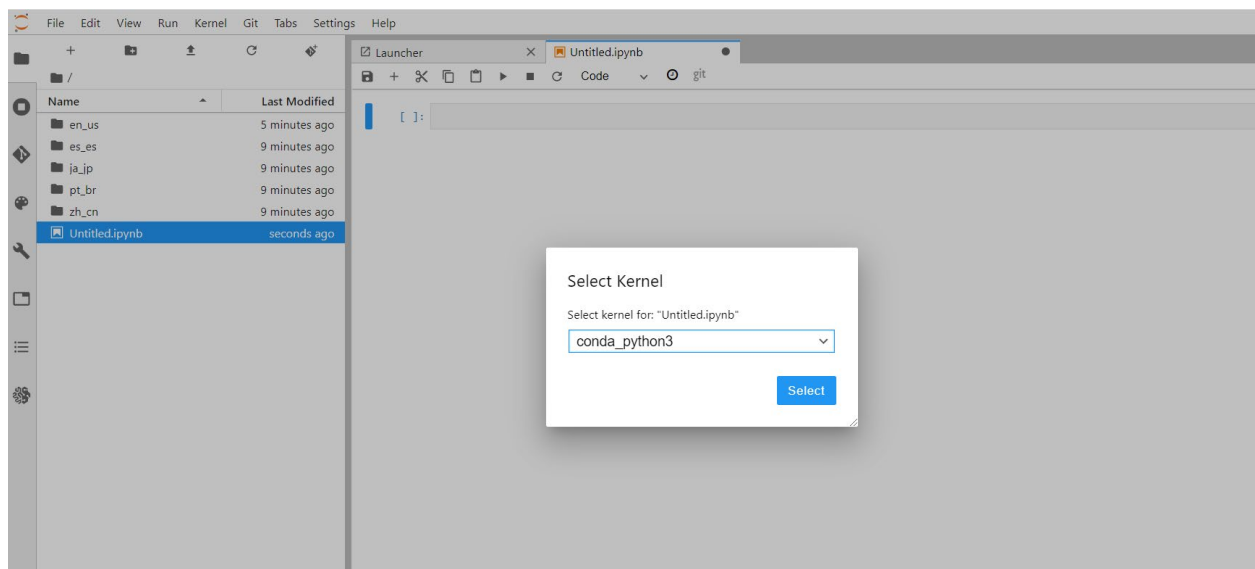
Jupyter Lab for the Notebook



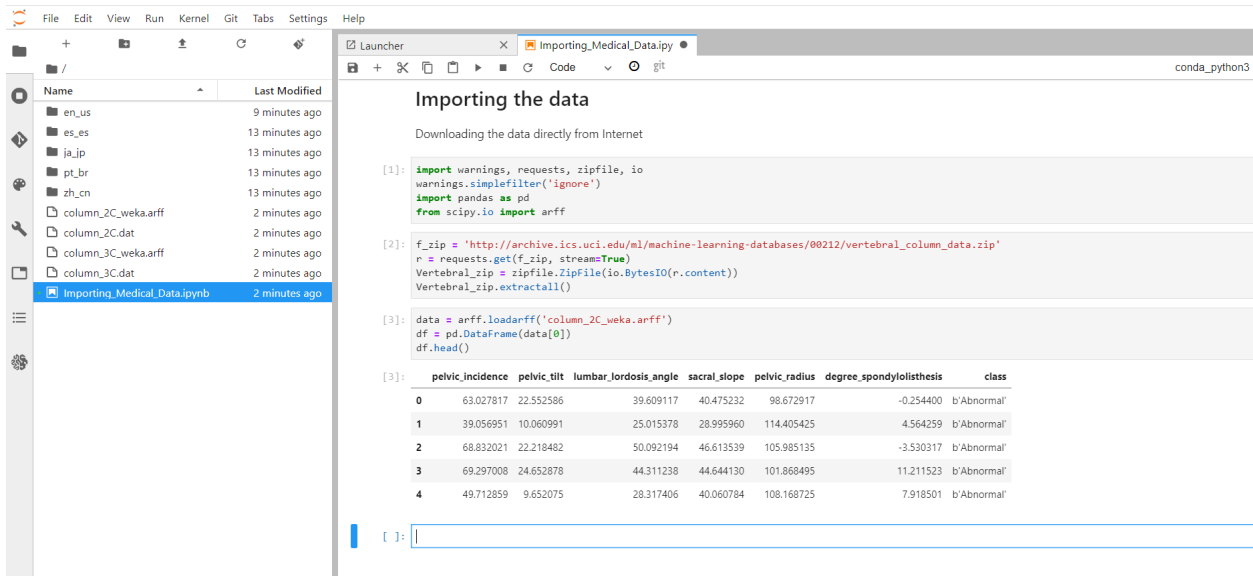
Ipynb files opens in a new window in Jupyter Lab



Creating New Notebook

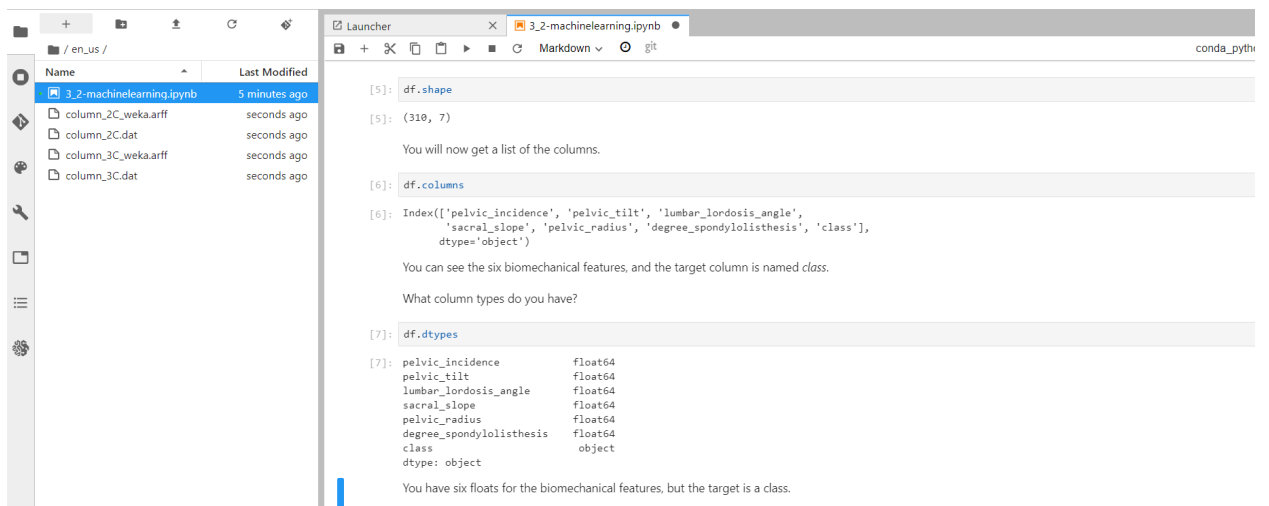


Selecting the Kernel for New Notebook



Working on New Notebook (Downloading and Importing Data)

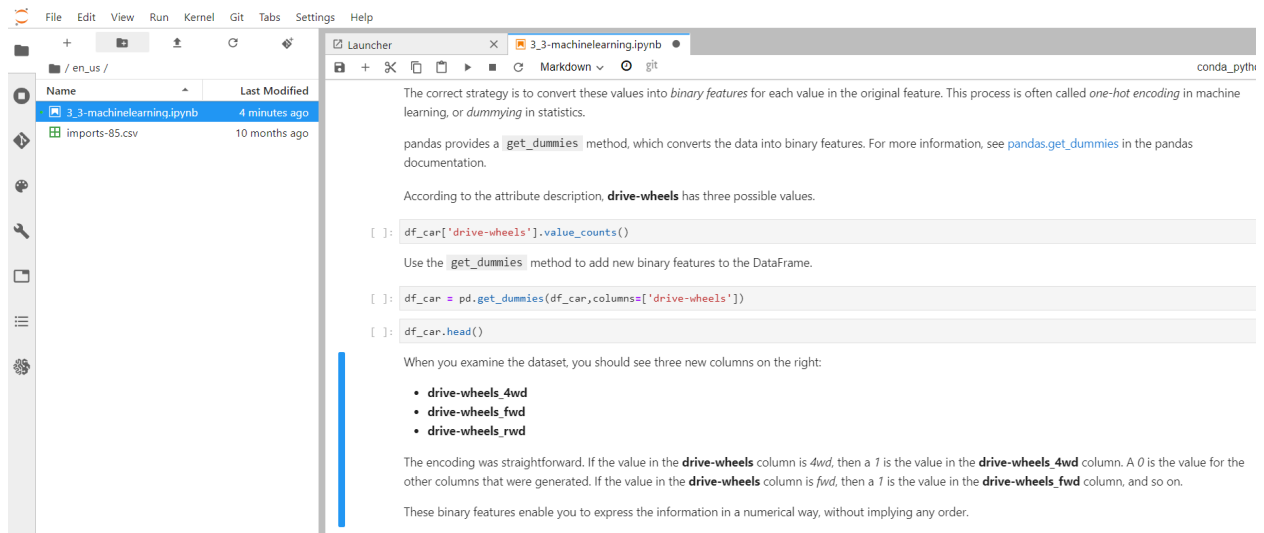
2. Exploring Data using Pandas, Matplotlib



Exploratory Data Analysis

3. Feature Engineering

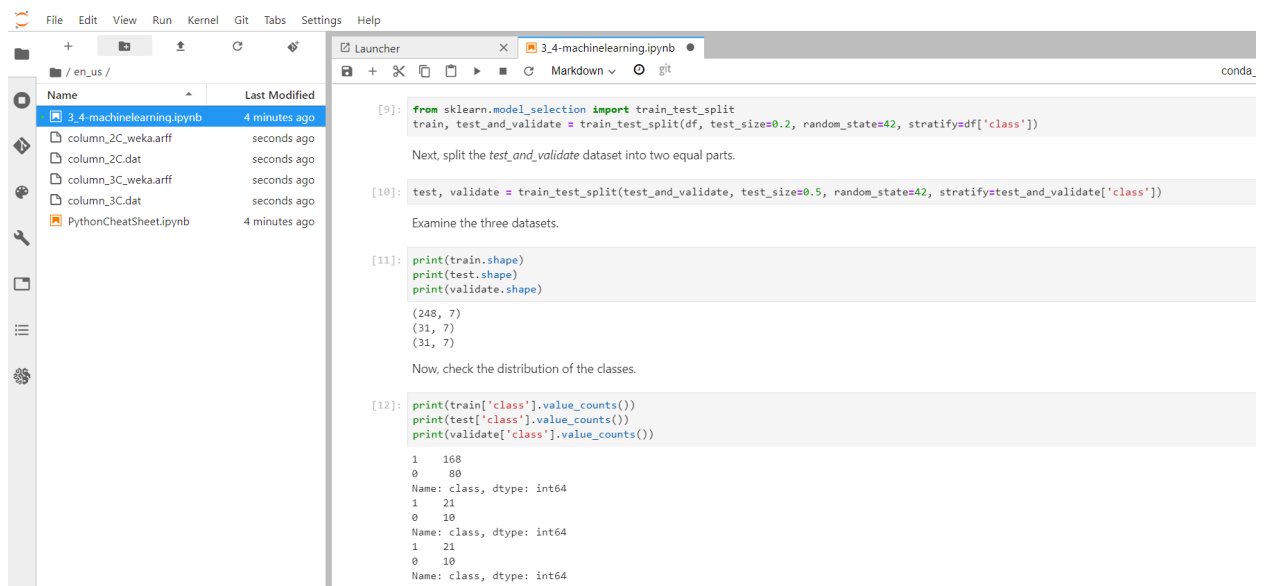
- Cleaning Data
- Dealing with Outliers and Selecting Features
- Encoding Categorical Data



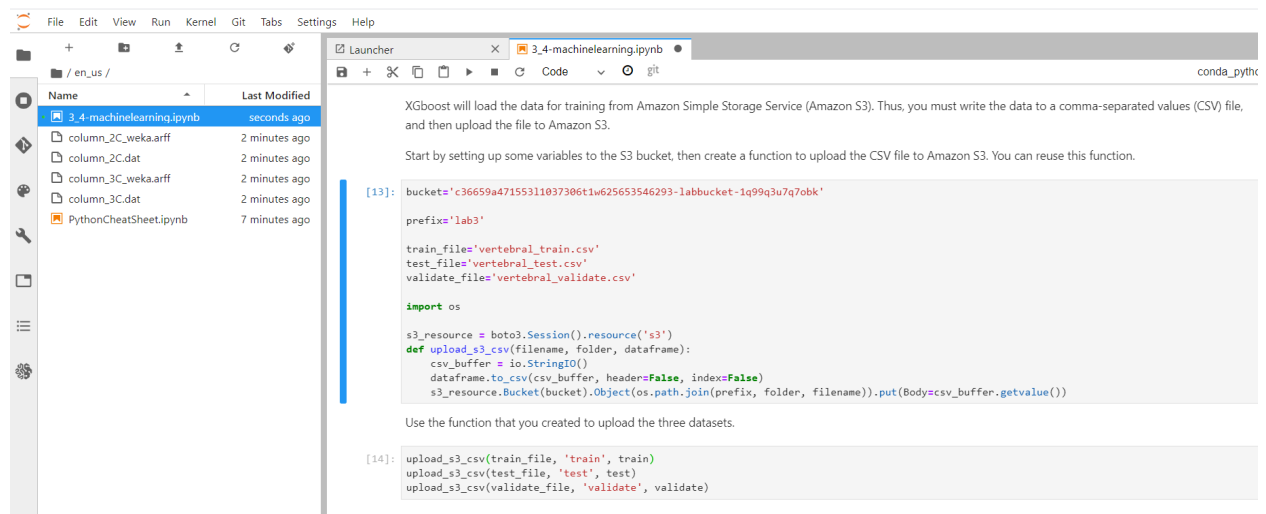
Encoding Categorical Data using Pandas (get_dummies) – AWS SageMaker Notebook

4. Training

- Train / Test / Validation Split



- Uploading Data to S3 Bucket for Model Building



The screenshot shows the JupyterLab interface with a file browser on the left and a code editor on the right. The file browser shows a directory structure with files like 'column_2C_weka.arff', 'column_2C.dat', 'column_3C_weka.arff', 'column_3C.dat', and 'PythonCheatSheet.ipynb'. The code editor shows the following code:

```
[13]: bucket='c36659a47155311037306t1w625653546293-labbucket-1q99q3u7q7obk'

prefix='lab3'

train_file='vertebral_train.csv'
test_file='vertebral_test.csv'
validate_file='vertebral_validate.csv'

import os

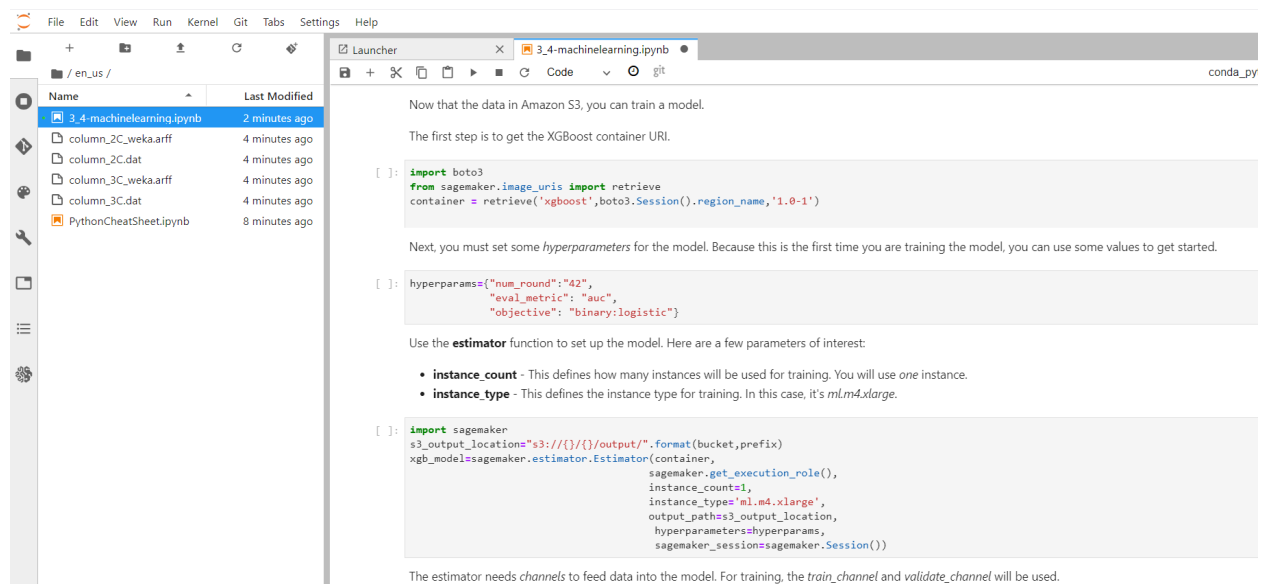
s3_resource = boto3.Session().resource('s3')
def upload_s3_csv(filename, folder, dataframe):
    csv_buffer = io.StringIO()
    dataframe.to_csv(csv_buffer, header=False, index=False)
    s3_resource.Bucket(bucket).Object(os.path.join(prefix, folder, filename)).put(Body=csv_buffer.getvalue())

Use the function that you created to upload the three datasets.

[14]: upload_s3_csv(train_file, 'train', train)
upload_s3_csv(test_file, 'test', test)
upload_s3_csv(validate_file, 'validate', validate)
```

XGBoost Model loads data from S3 bucket (uploading data to S3 bucket)

- Training the Model



The screenshot shows the JupyterLab interface with a file browser on the left and a code editor on the right. The file browser shows a directory structure with files like 'column_2C_weka.arff', 'column_2C.dat', 'column_3C_weka.arff', 'column_3C.dat', and 'PythonCheatSheet.ipynb'. The code editor shows the following code:

```
Now that the data in Amazon S3, you can train a model.

The first step is to get the XGBoost container URL.

[ ]: import boto3
from sagemaker.image_uris import retrieve
container = retrieve('xgboost', boto3.Session().region_name, '1.0-1')

Next, you must set some hyperparameters for the model. Because this is the first time you are training the model, you can use some values to get started.

[ ]: hyperparams={"num_round": "42",
                 "eval_metric": "auc",
                 "objective": "binary:logistic"}

Use the estimator function to set up the model. Here are a few parameters of interest:



- instance_count - This defines how many instances will be used for training. You will use one instance.
- instance_type - This defines the instance type for training. In this case, it's ml.m4.xlarge.



[ ]: import sagemaker
s3_output_location=s3://[ ]/[ ]/output/.format(bucket, prefix)
xgb_model=sagemaker.estimator.Estimator(container,
sagemaker.get_execution_role(),
instance_count=1,
instance_type='ml.m4.xlarge',
output_path=s3_output_location,
hyperparameters=hyperparams,
sagemaker_session=sagemaker.Session())

The estimator needs channels to feed data into the model. For training, the train_channel and validate_channel will be used.
```

XGBoost Model Retrieved from Container and setting up

- Fitting the Model

The screenshot displays a JupyterLab environment. On the left, a file browser shows the directory structure with files: `3.4-machinelearning.ipynb` (2 minutes ago), `column_2C_weka.arff` (6 minutes ago), `column_2C.dat` (6 minutes ago), `column_3C_weka.arff` (6 minutes ago), `column_3C.dat` (6 minutes ago), and `PythonCheatSheet.ipynb` (10 minutes ago). The main area is a code editor for `3.4-machinelearning.ipynb`. It contains the following code:

```
[17]: import sagemaker
s3_output_location=s3://()/output/".format(bucket,prefix)
xgb_model=sagemaker.estimator.Estimator(container,
                                         sagemaker.get_execution_role(),
                                         instance_count=1,
                                         instance_type='ml.m4.xlarge',
                                         output_path=s3_output_location,
                                         hyperparameters=hyperparams,
                                         sagemaker_session=sagemaker.Session())
```

The estimator needs *channels* to feed data into the model. For training, the *train_channel* and *validate_channel* will be used.

```
[ ]: train_channel = sagemaker.inputs.TrainingInput(
      "s3://()/train/".format(bucket,prefix,train_file),
      content_type='text/csv')

validate_channel = sagemaker.inputs.TrainingInput(
      "s3://()/validate/".format(bucket,prefix,validate_file),
      content_type='text/csv')

data_channels = {'train': train_channel, 'validation': validate_channel}
```

Running `fit` will train the model.

Note: This process can take up to 5 minutes.

```
[ ]: xgb_model.fit(inputs=data_channels, logs=False)
```

After the training is complete, you are ready to test and evaluate the model. However, you will do testing and validation in later labs.

Creating Channels and Fitting the Model