DataRobot

DataRobot.Python.Cheat_Sheet

Connecting

◆ To start with DataRobot, first connect, the endpoint here is US Managed Cloud

```
dr.Client(token='<API TOKEN>',
          endpoint='https://app.datarobot.com/api/v2/')
```

Create Project

◆ Dataframe

```
project = dr.Project.create(project_name=PROJECT_NAME,
                            sourcedata=df)
```

◆ Dataset

```
dataset = Dataset.create_from_file(file_path=DATA_PATH)
project = Project.create_from_dataset(dataset.id)
```

◆ URL

```
project = create_from_url(url, do_snapshot=False)
```

◆ Start Binary Classification, Regression, Multi-class Project

AutoML

```
project.set_target(target=TARGET, mode='quick',
                   metric=None, worker_count=-1,
                   partitioning_method=None,
                   advanced_options=None)
```

◆ Create Calendar File

AutoTS

```
CAL = dr.CalendarFile.create('Calendar.csv',
    calendar_name='Holidays')
```

◆ DateTime Partitioning

```
TIME_PARTITION = dr.DatetimePartitioningSpecification(
    use_time_series=True, datetime_partition_column=DATE,
    multiseries_id_columns=None,
    forecast_window_start=None,
    forecast_window_end=None,
    calendar_id=CAL.id)
```

◆ Start Project

```
project.set_target(target=TARGET,
                   partitioning_method=TIME_PARTITION)
```

Feature Lists

◆ Get Feature Lists

```
project.get_modeling_featurelists()
```

◆ Create Feature Lists

```
project.create_modeling_featurelist('Model This',
                                    selected_features)
```

Blueprints

◆ List Blueprints

```
project.get_blueprints()
```

◆ Get Blueprint

```
BlueprintChart.get(project_id, blueprint_id)
```

◆ View Blueprint Chart

```
print(bp_chart.to_graphviz())
```

◆ Get Blueprint Documentation

```
bp = Blueprint.get(project_id, blueprint_id)
docs = bp.get_documents()
```

Models

◆ Get Models

```
project.get_models(project_id, model_id)
```

◆ Select Model

```
menu = project.get_blueprints()
blueprint = menu[0]
```

◆ Train new feature list or sample size

```
model.train(featurelist_id=custom_featurelist.id)
```

◆ Train for a new duration

```
DURATION = dr.helpers.partitioning_methods \
    .construct_duration_string(years=17,
                               months=5,
                               days=1)
model.train_datetime(featurelist_id=fl_id,
                     training_duration=DURATION)
```

Evaluating

◆ Lift Chart

```
model.get_lift_chart(source='validation')
```

◆ ROC/Curve

```
model.get_roc_curve('validation')
```

◆ Residuals

```
ExternalResidualsChart.get(project_id,
                           model_id,
                           dataset_id)
```

◆ Confusion Matrix

```
model.get_confusion_chart(source='validation')
```

◆ Profit

```
dr.PayoffMatrix.create(project_id,
                       name,
                       true_positive_value=100,
                       true_negative_value=10,
                       false_positive_value=0,
                       false_negative_value=-10)
```

Understanding

◆ Feature Impact

```
model.get_or_request_feature_impact()
```

◆ Prediction Explanations

```
dr.PredictionExplanationsInitialization.create(
    project.id,
    model.id)
dr.PredictionExplanations.create(
    project.id,
    model.id,
    dataset.id)
pe_job.get_result_when_complete()
```

◆ Feature Effects

```
model.request_feature_effect()
```

◆ Word Cloud

```
model.get_word_cloud()
```

Advanced Tuning

◆ Interactive Tuning Interface

```
tune = model.start_advanced_tuning_session()
```

◆ Parameters available for Tuning

```
tune.get_task_names()
```

◆ Set Tuning Parameters

```
tune.set_parameter(task_name=task_name,
                   parameter_name='EUREQA_building_block__sine',
                   value=1)
```

◆ Run Tuned Model

```
job = tune.run()
new_model = job.get_result_when_complete()
```

Predictions

◆ List Prediction Servers

```
dr.PredictionServer.list()
```

◆ Create a Deployment

```
deployment = dr.Deployment.create_from_learning_model(
    model.id,
    label='New Deployment',
    description='A new deployment',
    default_prediction_server_id=prediction_server.id)
```

◆ List Deployments

```
dr.Deployment.list()
```

◆ Get Deployment

```
dr.Deployment.get(deployment_id='5c939e08962')
```

◆ Batch Predictions

```
dr.BatchPredictionJob.score_to_file(deployment_id,
                                    input_file,
                                    output_file)
```

◆ Model Replacement

```
deployment.replace_model('5c0a969859b',
                         MODEL_REPLACEMENT_REASON.ACCURACY)
```

◆ Drift Tracking

```
deployment.get_target_drift()
```

◆ Predictions from a Deployment

```
predictions_response = requests.post(PREDICTION_ENDPOINT,
                                     data=PAYLOAD,
                                     headers=headers)
```

Further Resources

◆ Python documentation

https://datarobot-public-api-client.readthedocs-hosted.com

◆ DataRobot Github Community

https://github.com/datarobot-community

◆ Ask Questions

https://community.datarobot.com