Model Wrapping with Modeva :: CHEATSHEET

Modeva-AI

Model Wrapping enables seamless integration of diverse predictive models into Modeva framework, including but not limited to


```
from modeva.models import
    MoRegressor, MoClassifier,
    MoSKLearnRegressor,
    MoSKLearnClassifier,
    MoScoredRegressor,
    MoScoredClassifier
```

Scikit-learn Model Wrapper

Example: Sklearn Lasso regressor

```
from sklearn.linear_model import Lasso
model = MoSKLearnRegressor(name="str", estimator
    =Lasso(alpha=0.1))
```

Example: LightGBM classifier

```
from lightgbm import LGBMClassifier
model = MoSKLearnClassifier(name="str",
    estimator=LGBMClassifier(verbose=-1))
```

Scored Model Wrappers

It requires creating a Modeva dataset with prediction scores, w/o need of model objects.

```
ds.set_prediction(feature="pred")
MoScoredRegressor(dataset=ds)
ds.set_prediction_proba(feature="pred_proba")
MoScoredClassifier(dataset=ds)
```

Arbitrary Model Wrappers

It requires defining predict_function and predict_proba_function. Examples on the right.

```
MoRegressor(name="str", predict_function=fun1)
MoClassifier(name="str", predict_function=fun1,
    predict_proba_function=fun2)
```

Examples of Wrapping Arbitrary Models

```
from pyspark.ml.feature import VectorAssembler
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("str").getOrCreate()
def fun1(X):
  X_spark = spark.createDataFrame(X)
  X_trans = VectorAssembler(...).transform(X_spark)
  res = spark_model.transform(X_trans).select("pred")
  return np.array([row.pred for row in res.collect()])
model = MoRegressor(name="str", predict_function=fun1)
```

```
import h2o
h2o.init()
def fun1(X):
    X_h2o = h2o.H2OFrame(X)
    res = h2o_model.predict(X_h2o)["predict"]
    return res.as_data_frame().values.flatten()
model = MoRegressor(name="str", predict_function=fun1)
```

```
import requests
REST_API = "https://REST-API-endpoint/predict"
def fun1(X):
    payload = {"inputs": X.to_dict(orient="records")}
    response = requests.post(REST_API, json=payload)
    response.raise_for_status()
    return np.array(response.json()["prediction"])
model = MoRegressor(name="str", predict_function=fun1)
```