Highlight Note

Interpretability During Inferencing

In some scenarios, you might want to generate explanations along with predictions from a published model.

Register a Scoring Explainer with the Model

The first step in this process is to create a *scoring explainer* as a wrapper for your model explainer, and register it in the same workspace as your model.

```
from interpret.ext.blackbox import TabularExplainer
from azureml.interpret.scoring.scoring_explainer import KernelScoringExplainer, save
from azureml.core import Model

# Get a registered model
loan_model - ws.models['loan_model']

tab_explainer = TabularExplainer(model = loan_model,
                            initialization_examples=X_test,
                            features=['loan_amount','income','age','marital_status'],
                            classes=['reject', 'approve'])

# Create and save a scoring explainer
scoring_explainer = KernelScoringExplainer(tab_explainer, X_test[0:100])
save(scoring_explainer, directory='explainer', exist_ok=True)

# Register the explainer (like a model)
Model.register(ws, model_name='loan_explainer', 'explainer/scoring_explainer.pkl')
```

Create a Scoring Script to Include Explanations

After registering the explainer, you can create a scoring script for a real-time service that loads the explainer and uses it to return explanations along with predictions.

```
import json
import joblib
from azureml.core.model import Model

# Called when the service is loaded
def init():
    global model, explainer
    # load the model
    model_path = Model.get_model_path('loan_model')
    model = joblib.load(model_path)
    # load the explainer
    explainer_path = Model.get_model_path('loan_explainer')
    explainer = joblib.load(explainer_path)

# Called when a request is received
def run(raw_data):
```

```
    # Get the input data
    data = np.array(json.loads(raw_data)['data'])
    # Get a prediction from the model
    predictions = model.predict(data)
    # Get explanations
    importance_values = explainer.explain(data)
    # Return the predictions and explanations as JSON
    return {"predictions":predictions.tolist(),"importance":importance_values}
```

Deploy the Inferencing Service

With the scoring script created, you can deploy the service - referencing both the predictive model and the explainer.

```
service = Model.deploy(ws, 'loan-svc', [model, explainer], inf_config, dep_config)
```

Retrieving Predictions and Explanations

When you consume the service, the JSON returned includes both the predictions and the associated local feature importance values:

```
import json

# New loan application data
x_new = [[55000, 3500, 37, 1]]
json_data = json.dumps({"data": x_new})
response = service.run(input_data = json_data)
print(response)
```

```
{
  'predictions': [1],
  'local_importance': [[[-0.12, -0.15, -0.03, 0.11]],
                       [[0.12, 0.15, 0.03, -0.11]]]
}
```