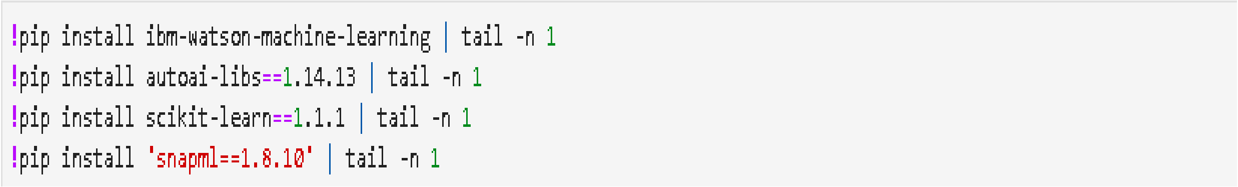
Machine Learning Model Deployment on IBM Cloud Watson Studio

Project title: Predictive analysis

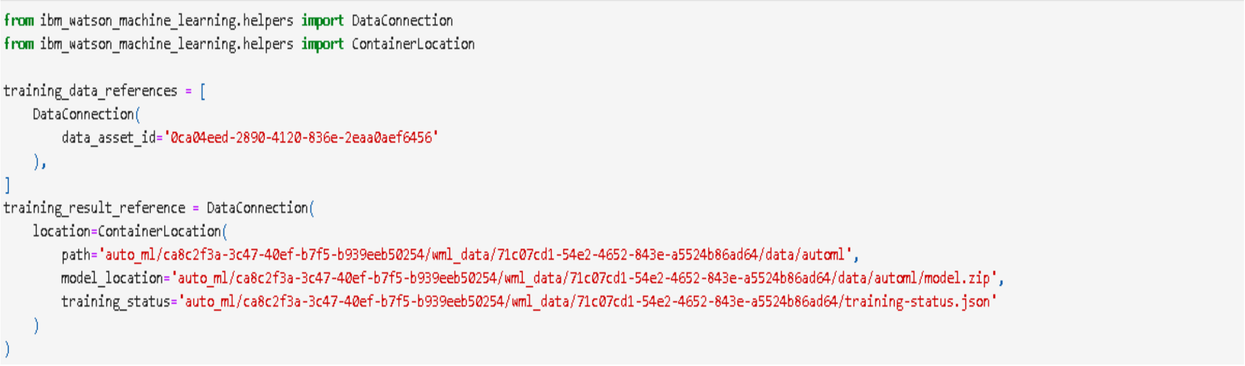
Setup

Package installation:



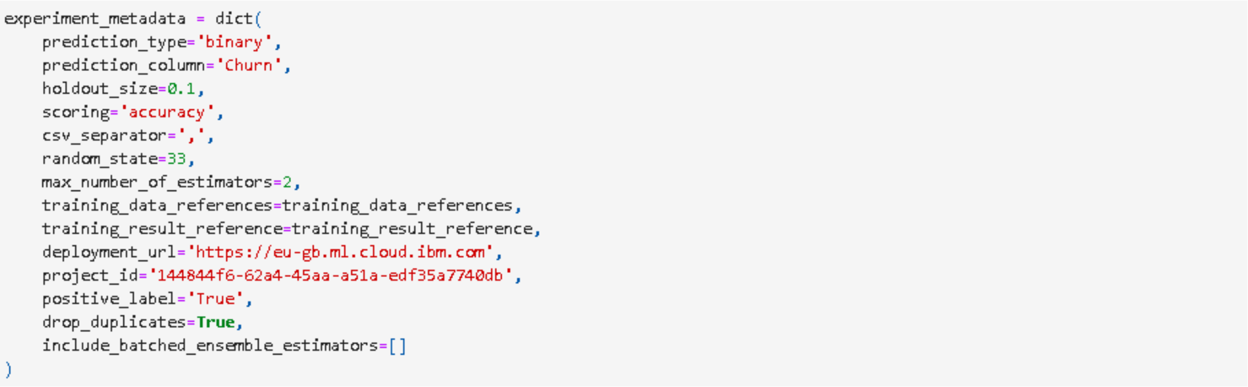
```
!pip install ibm-watson-machine-learning | tail -n 1
!pip install autoai-libs==1.14.13 | tail -n 1
!pip install scikit-learn==1.1.1 | tail -n 1
!pip install 'snapml==1.8.10' | tail -n 1
```

AutoAI experiment metadata:



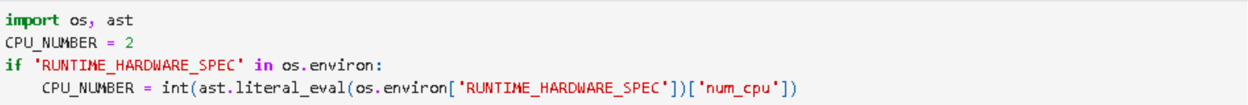
```
from ibm_watson_machine_learning.helpers import DataConnection
from ibm_watson_machine_learning.helpers import ContainerLocation

training_data_references = [
    DataConnection(
        data_asset_id='0ca04eed-2890-4120-836e-2eaa0aef6456'
    ),
]
training_result_reference = DataConnection(
    location=ContainerLocation(
        path='auto_ml/ca8c2f3a-3c47-40ef-b7f5-b939eeb50254/wml_data/71c07cd1-54e2-4652-843e-a5524b86ad64/data/automl',
        model_location='auto_ml/ca8c2f3a-3c47-40ef-b7f5-b939eeb50254/wml_data/71c07cd1-54e2-4652-843e-a5524b86ad64/data/automl/model.zip',
        training_status='auto_ml/ca8c2f3a-3c47-40ef-b7f5-b939eeb50254/wml_data/71c07cd1-54e2-4652-843e-a5524b86ad64/training-status.json'
    )
)
```

The following cell contains input parameters provided to run the AutoAI experiment in Watson Studio



```
experiment_metadata = dict(
    prediction_type='binary',
    prediction_column='Churn',
    holdout_size=0.1,
    scoring='accuracy',
    csv_separator=',',
    random_state=33,
    max_number_of_estimators=2,
    training_data_references=training_data_references,
    training_result_reference=training_result_reference,
    deployment_url='https://eu-gb.ml.cloud.ibm.com',
    project_id='144844f6-62a4-45aa-a51a-edf35a7740db',
    positive_label='True',
    drop_duplicates=True,
    include_batched_ensemble_estimators=[]
)
```

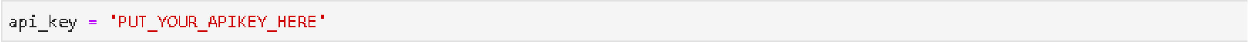
Set n_jobs parameter to the number of available CPUs



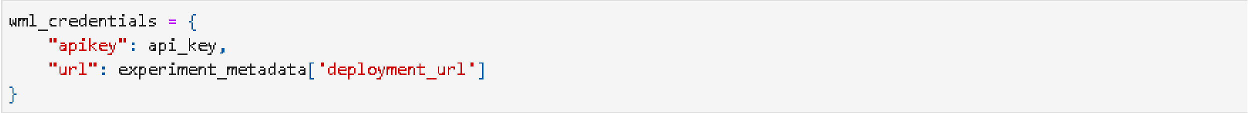
```
import os, ast
CPU_NUMBER = 2
if 'RUNTIME_HARDWARE_SPEC' in os.environ:
    CPU_NUMBER = int(ast.literal_eval(os.environ['RUNTIME_HARDWARE_SPEC'])['num_cpu'])
```

2. Watson Machine Learning connection

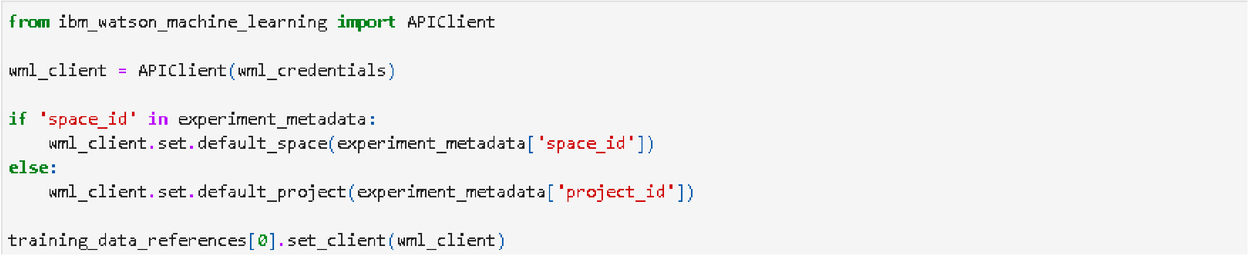
This cell defines the credentials required to work with the Watson Machine Learning service.



```
api_key = 'PUT_YOUR_APIKEY_HERE'
```



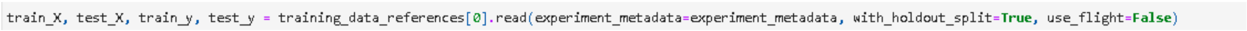
```
wml_credentials = {
    "apikey": api_key,
    "url": experiment_metadata['deployment_url']
}
```



```
from ibm_watson_machine_learning import APIClient

wml_client = APIClient(wml_credentials)

if 'space_id' in experiment_metadata:
    wml_client.set.default_space(experiment_metadata['space_id'])
else:
    wml_client.set.default_project(experiment_metadata['project_id'])

training_data_references[0].set_client(wml_client)
```

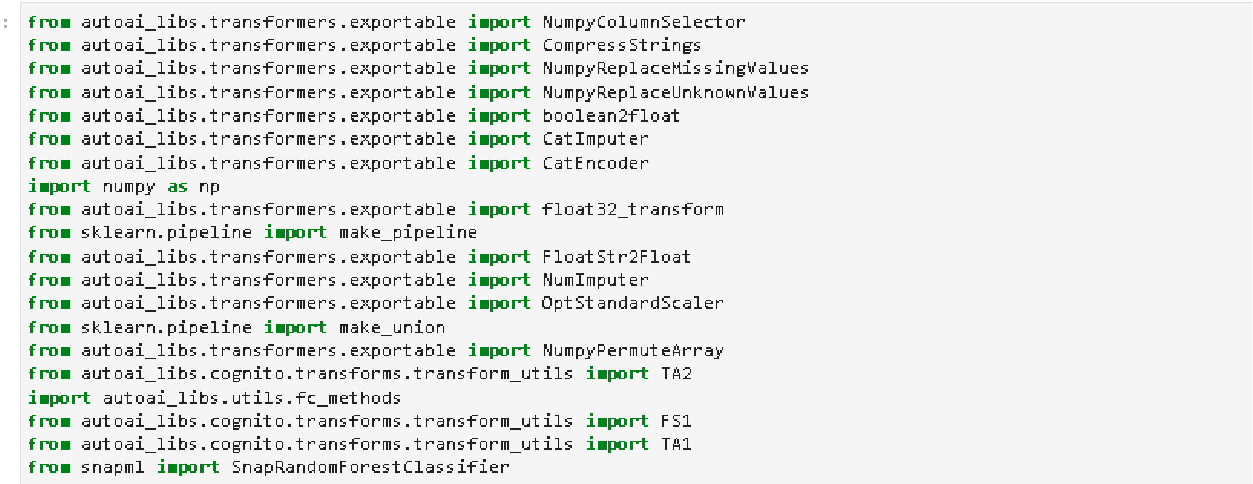
3. Pipeline inspection

Read training data



```
train_X, test_X, train_y, test_y = training_data_references[0].read(experiment_metadata=experiment_metadata, with_holdout_split=True, use_flight=False)
```

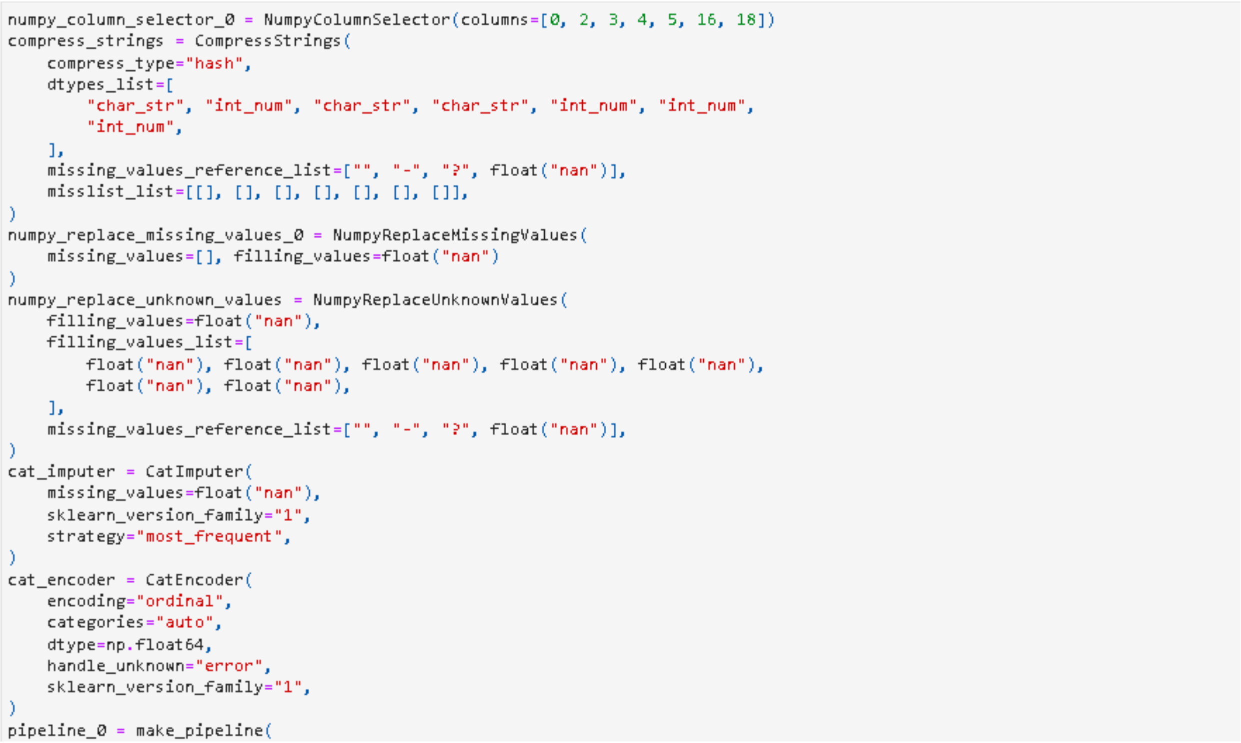
Create pipeline



```
from autoai_libs.transformers.exportable import NumpyColumnSelector
from autoai_libs.transformers.exportable import CompressStrings
from autoai_libs.transformers.exportable import NumpyReplaceMissingValues
from autoai_libs.transformers.exportable import NumpyReplaceUnknownValues
from autoai_libs.transformers.exportable import boolean2float
from autoai_libs.transformers.exportable import CatImputer
from autoai_libs.transformers.exportable import CatEncoder
import numpy as np
from autoai_libs.transformers.exportable import float32_transform
from sklearn.pipeline import make_pipeline
from autoai_libs.transformers.exportable import FloatStr2Float
from autoai_libs.transformers.exportable import NumImputer
from autoai_libs.transformers.exportable import OptStandardScaler
from sklearn.pipeline import make_union
from autoai_libs.transformers.exportable import NumpyPermuteArray
from autoai_libs.cognito.transforms.transform_utils import TA2
import autoai_libs.utils.fc_methods
from autoai_libs.cognito.transforms.transform_utils import FS1
from autoai_libs.cognito.transforms.transform_utils import TA1
from snapml import SnapRandomForestClassifier
```

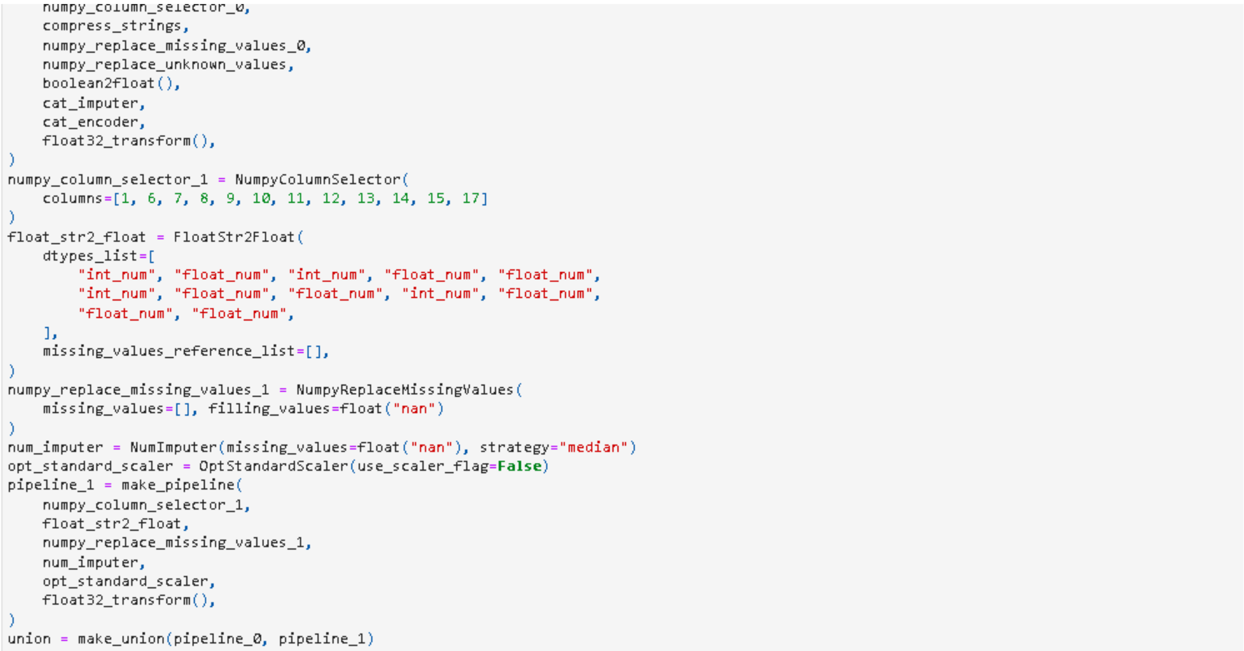
Pre-processing & Estimator.

1.



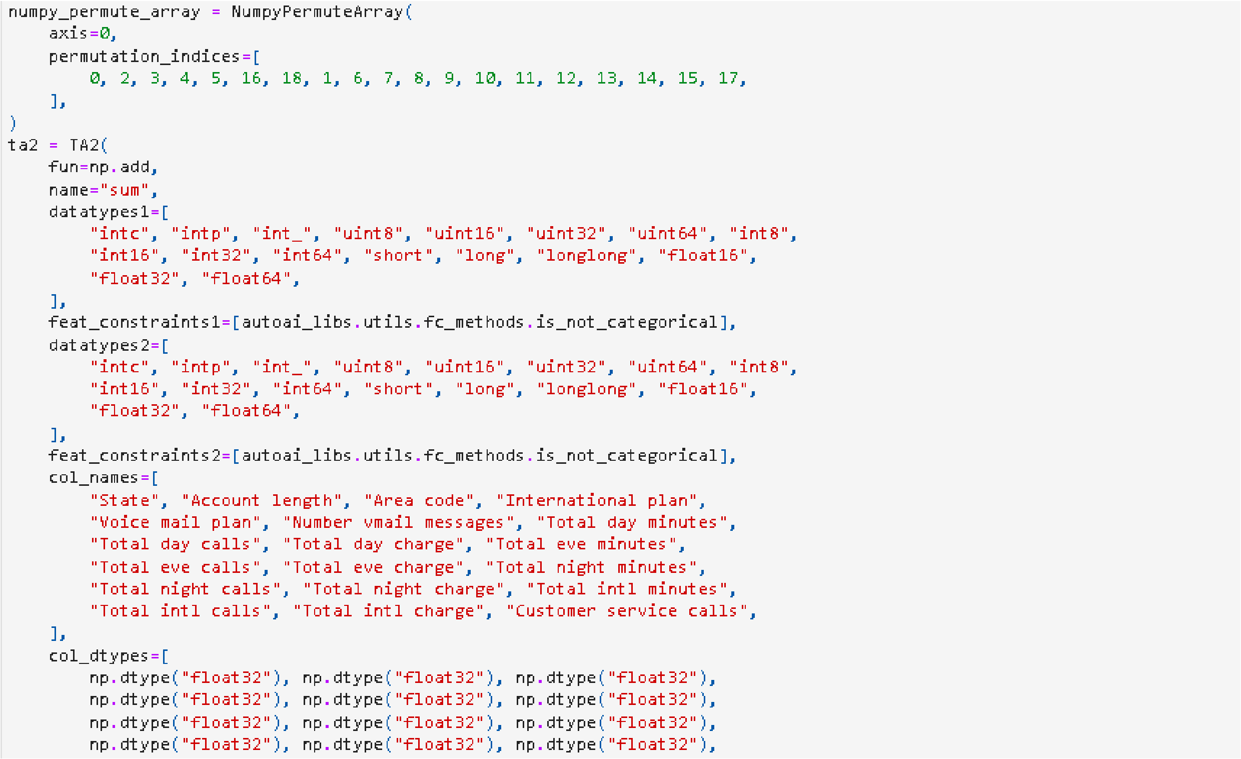
```
numpy_column_selector_0 = NumpyColumnSelector(columns=[0, 2, 3, 4, 5, 16, 18])
compress_strings = CompressStrings(
    compress_type="hash",
    dtypes_list=[
        "char_str", "int_num", "char_str", "char_str", "int_num", "int_num",
        "int_num",
    ],
    missing_values_reference_list=["", "-", "?", float("nan")],
    misslist_list=[[], [], [], [], [], [], []],
)
numpy_replace_missing_values_0 = NumpyReplaceMissingValues(
    missing_values=[], filling_values=float("nan")
)
numpy_replace_unknown_values = NumpyReplaceUnknownValues(
    filling_values=float("nan"),
    filling_values_list=[
        float("nan"), float("nan"), float("nan"), float("nan"), float("nan"),
        float("nan"), float("nan"),
    ],
    missing_values_reference_list=["", "-", "?", float("nan")],
)
cat_imputer = CatImputer(
    missing_values=float("nan"),
    sklearn_version_family="1",
    strategy="most_frequent",
)
cat_encoder = CatEncoder(
    encoding="ordinal",
    categories="auto",
    dtype=np.float64,
    handle_unknown="error",
    sklearn_version_family="1",
)
pipeline_0 = make_pipeline(
```

2.



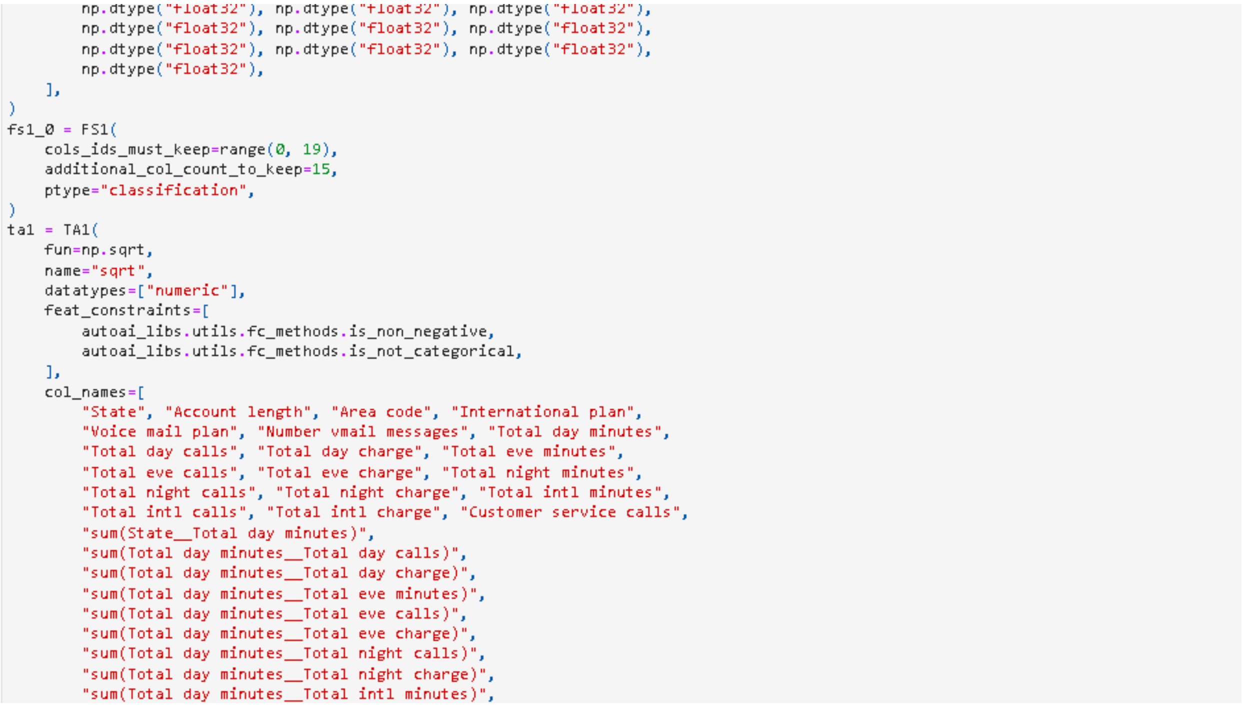
```
    numpy_column_selector_0,
    compress_strings,
    numpy_replace_missing_values_0,
    numpy_replace_unknown_values,
    boolean2float(),
    cat_imputer,
    cat_encoder,
    float32_transform(),
)
numpy_column_selector_1 = NumpyColumnSelector(
    columns=[1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17]
)
float_str2_float = FloatStr2Float(
    dtypes_list=[
        "int_num", "float_num", "int_num", "float_num", "float_num",
        "int_num", "float_num", "float_num", "int_num", "float_num",
        "float_num", "float_num",
    ],
    missing_values_reference_list=[],
)
numpy_replace_missing_values_1 = NumpyReplaceMissingValues(
    missing_values=[], filling_values=float("nan")
)
num_imputer = NumImputer(missing_values=float("nan"), strategy="median")
opt_standard_scaler = OptStandardScaler(use_scaler_flag=False)
pipeline_1 = make_pipeline(
    numpy_column_selector_1,
    float_str2_float,
    numpy_replace_missing_values_1,
    num_imputer,
    opt_standard_scaler,
    float32_transform(),
)
union = make_union(pipeline_0, pipeline_1)
```

3.



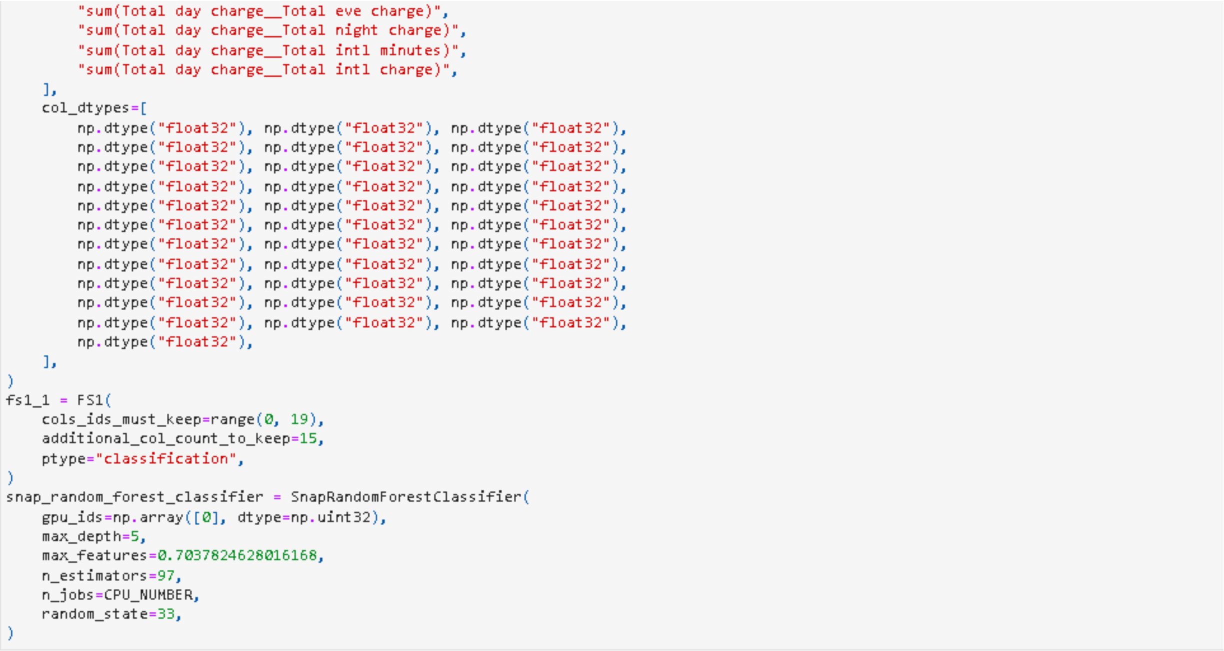
```
numpy_permute_array = NumpyPermuteArray(
    axis=0,
    permutation_indices=[
        0, 2, 3, 4, 5, 16, 18, 1, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17,
    ],
)
ta2 = TA2(
    fun=np.add,
    name="sum",
    datatypes1=[
        "intc", "intp", "int_", "uint8", "uint16", "uint32", "uint64", "int8",
        "int16", "int32", "int64", "short", "long", "longlong", "float16",
        "float32", "float64",
    ],
    feat_constraints1=[autoai_libs.utils.fc_methods.is_not_categorical],
    datatypes2=[
        "intc", "intp", "int_", "uint8", "uint16", "uint32", "uint64", "int8",
        "int16", "int32", "int64", "short", "long", "longlong", "float16",
        "float32", "float64",
    ],
    feat_constraints2=[autoai_libs.utils.fc_methods.is_not_categorical],
    col_names=[
        "State", "Account length", "Area code", "International plan",
        "Voice mail plan", "Number vmail messages", "Total day minutes",
        "Total day calls", "Total day charge", "Total eve minutes",
        "Total eve calls", "Total eve charge", "Total night minutes",
        "Total night calls", "Total night charge", "Total intl minutes",
        "Total intl calls", "Total intl charge", "Customer service calls",
    ],
    col_dtypes=[
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
```

4.



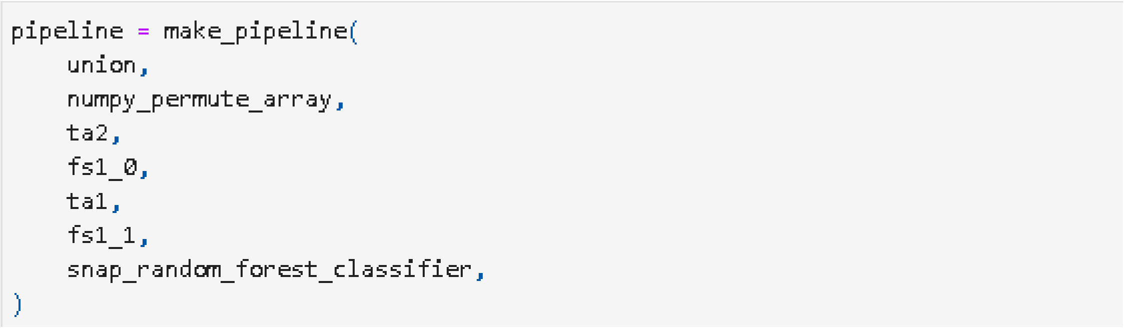
```
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"),
    ],
)
fs1_0 = FS1(
    cols_ids_must_keep=range(0, 19),
    additional_col_count_to_keep=15,
    ptype="classification",
)
ta1 = TA1(
    fun=np.sqrt,
    name="sqrt",
    datatypes=["numeric"],
    feat_constraints=[
        autoai_libs.utils.fc_methods.is_non_negative,
        autoai_libs.utils.fc_methods.is_not_categorical,
    ],
    col_names=[
        "State", "Account length", "Area code", "International plan",
        "Voice mail plan", "Number vmail messages", "Total day minutes",
        "Total day calls", "Total day charge", "Total eve minutes",
        "Total eve calls", "Total eve charge", "Total night minutes",
        "Total night calls", "Total night charge", "Total intl minutes",
        "Total intl calls", "Total intl charge", "Customer service calls",
        "sum(State__Total day minutes)",
        "sum(Total day minutes__Total day calls)",
        "sum(Total day minutes__Total day charge)",
        "sum(Total day minutes__Total eve minutes)",
        "sum(Total day minutes__Total eve calls)",
        "sum(Total day minutes__Total eve charge)",
        "sum(Total day minutes__Total night calls)",
        "sum(Total day minutes__Total night charge)",
        "sum(Total day minutes__Total intl minutes)",
```

5.



```
        "sum(Total day charge__Total eve charge)",
        "sum(Total day charge__Total night charge)",
        "sum(Total day charge__Total intl minutes)",
        "sum(Total day charge__Total intl charge)",
    ],
    col_dtypes=[
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"), np.dtype("float32"), np.dtype("float32"),
        np.dtype("float32"),
    ],
)
fs1_1 = FS1(
    cols_ids_must_keep=range(0, 19),
    additional_col_count_to_keep=15,
    ptype="classification",
)
snap_random_forest_classifier = SnapRandomForestClassifier(
    gpu_ids=np.array([0], dtype=np.uint32),
    max_depth=5,
    max_features=0.7037824628016168,
    n_estimators=97,
    n_jobs=CPU_NUMBER,
    random_state=33,
)
```

4. Pipeline:

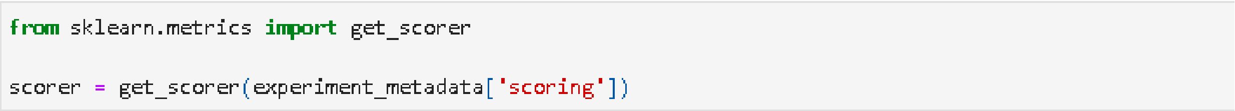


```
pipeline = make_pipeline(
    union,
    numpy_permute_array,
    ta2,
    fs1_0,
    ta1,
    fs1_1,
    snap_random_forest_classifier,
)
```

Train pipeline model

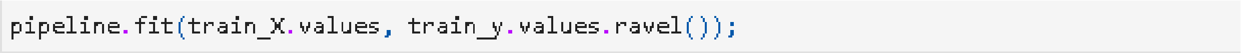
Define scorer from the optimization metric

This cell constructs the cell scorer based on the experiment metadata.



```
from sklearn.metrics import get_scorer

scorer = get_scorer(experiment_metadata['scoring'])
```

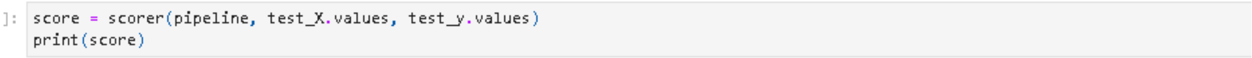
Fit pipeline model



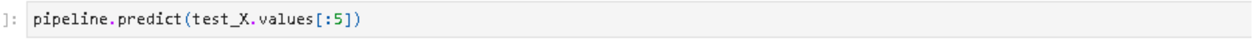
```
pipeline.fit(train_X.values, train_y.values.ravel());
```

5.Test pipeline model

Score the fitted pipeline with the generated scorer using the holdout dataset.



```
]: score = scorer(pipeline, test_X.values, test_y.values)
   print(score)
```



```
]: pipeline.predict(test_X.values[:5])
```

Store the model

```
model_metadata = {
    wml_client.repository.ModelMetaNames.NAME: 'Trained AutoAI pipeline'
}

stored_model_details = wml_client.repository.store_model(model=pipeline, meta_props=model_metadata, experiment_metadata=experiment_metadata)
```

Inspect the stored model details.

```
stored_model_details
```