Freedom.
Flexibility.
Cutting-edge AI.

Create models faster than ever with SAS Viya Workbench

Lars Arne Skår, Principal Pre-Sales Solutions Architect • Customer Advisory Technology Northern Europe
Pia Rønnevik, Customer Success Manager, FANS, SAS Institute

Copyright © SAS Institute Inc. All rights reserved.

Development Challenges

Systems, People and Processes

Data Access

Immediate access to data, validating data, and preparing for development.

Speed of Development

Quickly spin up, reuse existing code, and choose your coding language.

Seamless Deployment

Preparing for and seamlessly deploying models quickly and efficiently.

Flexible Infrastructure

Provisioning the right-sized environment and scaling up & down as needed.

Minimizing Costs

Optimizing and reducing cost and complexity within your workspace.

Copyright © SAS Institute Inc. All rights reserved.

SAS Viya Workbench

Purpose-built for
developers and modelers

Empower data science teams with freedom, flexibility, and cutting-edge analytics to develop and prepare models fast.

- Develop models with speed while optimizing infrastructure costs
- Accelerate team productivity for SAS & Python developers alike
- Access industry-leading analytics from a partner you trust

Access data. Build models. Prepare for deployment. All with Viya Workbench.

Copyright © SAS Institute Inc. All rights reserved.

Microsoft Teams

Demo FANS Mini Forum

2024-11-06 08:45 UTC

Recorded by

Pia Rønnevik

Organized by

Pia Rønnevik

Python-based modeling – open source or SAS, choice is yours

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

Predict Loan Default

This project shows how we can write Python in SAS Viya Workbench, and how we can substitute Python models using Sci-Kit Learn for SAS models using the sasviya.ml package

Load Dependencies

```
import numpy as np
import pandas as pd

# Packages for Building Model Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics

from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from sklearn.tree import DecisionTreeClassifier
#from sasviya.ml.tree import DecisionTreeClassifier
```

[162] ✓ 0.0s

Training with larger datasets – logreg and gradient boosting

Scitkit learn and sasviyaml libraries

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

Improving Credit Risk Scorecards with GenAI-based synthetic data

---

The demo is as follows:

- Load credit data
- Explore it
- Perform binning/transformations
- Build models using scikit-learn and SAS logistic regression methods
- Create a new SAS logistic regression model on synthetic data
- Register to Model Manager

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
%matplotlib inline

from sasviya.core import Action, Datalib, Table

def test():
    return print("test")

def woe_binning(input_data: pd.DataFrame, nominal_inputs: list[str], cont_inputs: list[str], target: str):
    act_transform = Action("dataPreprocess", "transform")
    # try: d
```

Make the models available to our Enterprise Viya environment

+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···

Register Model to Model Manager

+ Code + Markdown

```
from sasctl import pzmm
from sasctl import Session
from sasctl.services import model_repository as mr, model_management as mm
from pathlib import Path
import requests
import os
```

[ ]

```
## get access token for viya env using refresh token. change to your own viya server and preferred authentication method.
url = "https://viya-cauki.unx.sas.com/"
auth_url = f"{url}/SASLogon/oauth/token"
## reading long-lived refresh token from txt file
refresh_token = Path(f'/workspaces/{os.environ["DEFAULT_MOUNTNAME"]}/boost-credit-scorecard-performance/python/cauki_refresh_token.txt').read_text().replace('\n', '')
payload=f'grant_type=refresh_token&refresh_token={refresh_token}'
headers = {
  'Accept': 'application/json',
  'Content-Type': 'application/x-www-form-urlencoded',
  'Authorization': 'Basic c2FzLmNsaTo=',
}

response = requests.request("POST", auth_url, headers=headers, data=payload, verify=False)
access_token = response.json()['access_token']
```

[ ]

Copyright © SAS Institute Inc. All rights reserved.