

```
! nvidia-smi
```

```
Sat May 21 09:45:17 2022
```

NVIDIA-SMI		460.32.03		Driver Version: 460.32.03			CUDA Version: 11.2		

GPU	Name		Persistence-M		Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap		Memory-Usage		GPU-Util	Compute	M.
								MIG	M.
=====									
0	Tesla	T4	Off		00000000:00:04.0 Off				0
N/A	38C	P8	11W / 70W		0MiB / 15109MiB		0%	Default	
									N/A

Processes:												
GPU	GI	CI	PID	Type	Process name			GPU Memory				
	ID	ID						Usage				
=====												
No running processes found												

```
!pip install pytorch-tabnet optuna
```

```

  Downloading pytorch_tabnet-3.1.1-py3-none-any.whl (39 kB)
Collecting optuna
  Downloading optuna-2.10.0-py3-none-any.whl (308 kB)
    |████████████████████████████████████████| 308 kB 13.3 MB/s
Requirement already satisfied: numpy<2.0,>=1.17 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: scikit_learn>0.21 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: tqdm<5.0,>=4.36 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: torch<2.0,>=1.2 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: scipy>1.4 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Requirement already satisfied: sqlalchemy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from pytorch-tabnet)
Collecting cliff
  Downloading cliff-3.10.1-py3-none-any.whl (81 kB)
    |████████████████████████████████████████| 81 kB 13.3 MB/s
Collecting colorlog
  Downloading colorlog-6.6.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (from colorlog)
Collecting alembic
  Downloading alembic-1.7.7-py3-none-any.whl (210 kB)
    |████████████████████████████████████████| 210 kB 68.9 MB/s
Collecting cmaes>=0.8.2
  Downloading cmaes-0.8.2-py3-none-any.whl (15 kB)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from cmaes)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from cmaes)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from cmaes)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.7/dist-packages (from cmaes)
Collecting Mako
  Downloading Mako-1.2.0-py3-none-any.whl (78 kB)
    |████████████████████████████████████████| 78 kB 8.5 MB/s
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.7/dist-packages (from Mako)
Requirement already satisfied: PrettyTable>=0.7.2 in /usr/local/lib/python3.7/dist-packages (from Mako)

```

```

Collecting autopage>=0.4.0
  Downloading autopage-0.5.0-py3-none-any.whl (29 kB)
Collecting stevedore>=2.0.1
  Downloading stevedore-3.5.0-py3-none-any.whl (49 kB)
|████████████████████████████████████████| 49 kB 8.8 MB/s
Collecting cmd2>=1.0.0
  Downloading cmd2-2.4.1-py3-none-any.whl (146 kB)
|████████████████████████████████████████| 146 kB 68.0 MB/s
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.9.0-py2.py3-none-any.whl (112 kB)
|████████████████████████████████████████| 112 kB 70.8 MB/s
Collecting pyperclip>=1.6
  Downloading pyperclip-1.8.2.tar.gz (20 kB)
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: pyperclip
  Building wheel for pyperclip (setup.py) ... done
  Created wheel for pyperclip: filename=pyperclip-1.8.2-py3-none-any.whl size=1113
  Stored in directory: /root/.cache/pip/wheels/9f/18/84/8f69f8b08169c7bae2dde6bd7d
Successfully built pyperclip
Installing collected packages: pyperclip, pbr, stevedore, Mako, cmd2, autopage, cc
Successfully installed Mako-1.2.0 alembic-1.7.7 autopage-0.5.0 cliff-3.10.1 cmaes-

```

```

import pandas as pd
import numpy as np
from pytorch_tabnet.tab_model import TabNetClassifier
from sklearn.metrics import accuracy_score, classification_report
import optuna as opt
import torch
import os
import joblib

def make_save_cv_model(i,model_name,model,best_params,optim,output_path="./drive/MyDrive/S

''' This function saves cross validation model in the corresponding directory ( if the

if os.path.exists(os.path.join(output_path,f"{i}_{model_name}_{optim}")):
    joblib.dump(model, os.path.join(output_path,f"{i}_{model_name}_{optim}/{i}_model.z
    with open(os.path.join(output_path,f"{i}_{model_name}_{optim}/model_params.txt"),'
        file.write(str(best_params))
else:
    os.mkdir(os.path.join(output_path,f"{i}_{model_name}_{optim}"))
    joblib.dump(model, os.path.join(output_path,f"{i}_{model_name}_{optim}/{i}_model.z
    with open(os.path.join(output_path,f"{i}_{model_name}_{optim}/model_params.txt"),'
        file.write(str(best_params))

def train(fold_dict,fold,model_name,sc_df,tar_col,optim,optim_trial,k_folds=10,tar_cols=""

''' this function is used to train the model with parameters optimization using optuna

y = sc_df[tar_col]

```

```

x = sc_df.drop([tar_col],axis=1)
model_name = model_name
def objective(trial):
    train_index = fold_dict[fold]["train"]
    test_index = fold_dict[fold]["test"]
    clf = TabNetClassifier(n_d=trial.suggest_int("n_d", 8, 64),
                           n_a=trial.suggest_int("n_a", 8, 64),
                           n_steps = trial.suggest_int("n_steps",3,10),
                           gamma =trial.suggest_float("gamma", 1.0, 2.0),
                           n_independent = trial.suggest_int("n_independent",1,5),
                           n_shared = trial.suggest_int("n_shared",1,5),
                           momentum = trial.suggest_float("momentum", 0.01, 0.4),
                           optimizer_fn = torch.optim.Adam,
                           # scheduler_fn = torch.optim.lr_scheduler,
                           # scheduler_params = {"gamma" :trial.suggest_float("sch-gamm
                           verbose = verbose,
                           device_name = "auto"
                           )

    # print(f" train_index :: {train_index}")
    # print(f" test_index :: {test_index}")
    X_train,X_test = x.iloc[train_index,:], x.iloc[test_index,:]
    # print(X_train.shape, X_test.shape)
    X_train, X_test = X_train.to_numpy(dtype=np.float64), X_test.to_numpy(dtype=np.float
    Y_train, Y_test = y.iloc[train_index], y.iloc[test_index]
    Y_train, Y_test = Y_train.to_numpy(dtype=np.float64), Y_test.to_numpy(dtype=np.float
    print(Y_train.shape, Y_test.shape)
    clf.fit(X_train, Y_train,
            eval_set=[(X_test, Y_test)],
            eval_metric=['accuracy'])
    Y_pred = clf.predict(X_test)
    print(classification_report(Y_test, Y_pred, labels=[x for x in range(6)]))
    clf_report = classification_report(Y_test, Y_pred, labels=[x for x in range(6)])
    joblib.dump(clf_report,f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/classi
    with open(f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/classification_repo
    print(f"Saved classification_report at : ./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DAT
    acc = accuracy_score(Y_pred, Y_test)
    return acc

print(f"Starting optimization for fold : [{fold}]/[{k_folds}]")
study = opt.create_study(direction='maximize')
study.optimize(objective, n_trials=optim_trial)
best_params = study.best_params
print(f" Best params for fold : [{fold}]/[{k_folds}]")
print(best_params)
joblib.dump(best_params,f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/{model_
with open(f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/{model_name}/best_par
print(f"Saved best_params at : outputs/{model_name}/best_params/fold_{fold}_best_param
train_index = fold_dict[fold]["train"]
test_index = fold_dict[fold]["test"]
X_train,X_test = x.iloc[train_index,:], x.iloc[test_index,:]
# print(X_train.shape, X_test.shape)
X_train, X_test = X_train.to_numpy(dtype=np.float64), X_test.to_numpy(dtype=np.float64
Y_train, Y_test = y.iloc[train_index], y.iloc[test_index]
Y_train, Y_test = Y_train.to_numpy(dtype=np.float64), Y_test.to_numpy(dtype=np.float64
clf_model = TabNetClassifier(**study.best_params)

```

```

clf_model.fit(X_train,Y_train)
Y_pred = clf_model.predict(X_test)
accuracy = accuracy_score(Y_pred, Y_test)
with open(f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/{model_name}/{model_n
try:
    print("[++] Saving the model and parameters in corresponding directories")
    make_save_cv_model(fold,model_name,clf_model,best_params,optim=optim)
except:
    print("[-] Failed to save the model")

use_df = pd.read_csv("./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/data/trainable_
tar_col = "PCE_categorical"
model_name = "pytorch_tabnet"
optimizer = "Adam"
fold_dict = joblib.load("./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/inputs/fold_vals/fol
fold = 1

train(fold_dict = fold_dict,
      fold = fold,
      model_name=model_name,
      sc_df=use_df,
      tar_col=tar_col,
      optim=optimizer,
      optim_trial = 15)
print(f"[++] Ended the training process for fold {fold}")

epoch 45 | loss: 0.46047 | 0:01:50s
epoch 46 | loss: 0.46048 | 0:01:52s
epoch 47 | loss: 0.46125 | 0:01:55s
epoch 48 | loss: 0.45665 | 0:01:57s
epoch 49 | loss: 0.45394 | 0:02:00s
epoch 50 | loss: 0.44932 | 0:02:02s
epoch 51 | loss: 0.44796 | 0:02:04s
epoch 52 | loss: 0.45004 | 0:02:07s
epoch 53 | loss: 0.44179 | 0:02:09s
epoch 54 | loss: 0.43674 | 0:02:12s
epoch 55 | loss: 0.42961 | 0:02:14s
epoch 56 | loss: 0.43368 | 0:02:17s
epoch 57 | loss: 0.43521 | 0:02:19s
epoch 58 | loss: 0.42595 | 0:02:21s
epoch 59 | loss: 0.42035 | 0:02:24s
epoch 60 | loss: 0.42147 | 0:02:26s
epoch 61 | loss: 0.42219 | 0:02:28s
epoch 62 | loss: 0.41878 | 0:02:31s
epoch 63 | loss: 0.42434 | 0:02:33s
epoch 64 | loss: 0.41623 | 0:02:35s
epoch 65 | loss: 0.41559 | 0:02:38s
epoch 66 | loss: 0.41121 | 0:02:40s
epoch 67 | loss: 0.40493 | 0:02:43s
epoch 68 | loss: 0.40456 | 0:02:45s
epoch 69 | loss: 0.40205 | 0:02:47s
epoch 70 | loss: 0.40484 | 0:02:50s
epoch 71 | loss: 0.39602 | 0:02:52s
epoch 72 | loss: 0.40013 | 0:02:55s
epoch 73 | loss: 0.40166 | 0:02:57s
epoch 74 | loss: 0.39655 | 0:02:59s
epoch 75 | loss: 0.39222 | 0:03:02s

```

```
epoch 75 | loss: 0.3922 | 0:03:02s
epoch 76 | loss: 0.38811 | 0:03:04s
epoch 77 | loss: 0.38481 | 0:03:07s
epoch 78 | loss: 0.39306 | 0:03:09s
epoch 79 | loss: 0.40727 | 0:03:11s
epoch 80 | loss: 0.40324 | 0:03:14s
epoch 81 | loss: 0.39178 | 0:03:16s
epoch 82 | loss: 0.39285 | 0:03:18s
epoch 83 | loss: 0.40109 | 0:03:21s
epoch 84 | loss: 0.38447 | 0:03:23s
epoch 85 | loss: 0.38467 | 0:03:26s
epoch 86 | loss: 0.38094 | 0:03:28s
epoch 87 | loss: 0.37485 | 0:03:31s
epoch 88 | loss: 0.37232 | 0:03:33s
epoch 89 | loss: 0.38299 | 0:03:35s
epoch 90 | loss: 0.37776 | 0:03:38s
epoch 91 | loss: 0.37693 | 0:03:40s
epoch 92 | loss: 0.36476 | 0:03:43s
epoch 93 | loss: 0.36422 | 0:03:45s
epoch 94 | loss: 0.3783 | 0:03:47s
epoch 95 | loss: 0.36666 | 0:03:50s
epoch 96 | loss: 0.36485 | 0:03:52s
epoch 97 | loss: 0.36033 | 0:03:54s
epoch 98 | loss: 0.36102 | 0:03:57s
epoch 99 | loss: 0.3568 | 0:03:59s
[++] Saving the model and parameters in corresponding directories
[++] Ended the training process for fold 1
```

Fold 0 has started running on 20-05-22

Fold 0 has completed sucessfully on 17:00 20-05-22

Fold 1 has started running at 15:15 21-05-22

