```
! nvidia-smi
```

```
Mon May 23 12:50:18 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   56C    P8    10W /  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 20.8 MB/s
Requirement already satisfied: tqdm<5.0,>=4.36 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: numpy<2.0,>=1.17 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: scipy>1.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: torch<2.0,>=1.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: scikit_learn>0.21 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-
Collecting cmaes>=0.8.2
  Downloading cmaes-0.8.2-py3-none-any.whl (15 kB)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: sqlalchemy>=1.1.0 in /usr/local/lib/python3.7/dist-
Collecting colorlog
  Downloading colorlog-6.6.0-py2.py3-none-any.whl (11 kB)
Collecting cliff
  Downloading cliff-3.10.1-py3-none-any.whl (81 kB)
     |████████████████████████████████| 81 kB 8.2 MB/s
Collecting alembic
  Downloading alembic-1.7.7-py3-none-any.whl (210 kB)
     |████████████████████████████████| 210 kB 34.7 MB/s
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist
Collecting Mako
  Downloading Mako-1.2.0-py3-none-any.whl (78 kB)
     |████████████████████████████████| 78 kB 6.4 MB/s

Requirement already satisfied: importlib-resources in /usr/local/lib/python3.7/dis
Collecting cmd2>=1.0.0
```

```
  Downloading cmd2-2.4.1-py3-none-any.whl (146 kB)
     |████████████████████████████████| 146 kB 55.1 MB/s
Collecting autopage>=0.4.0
  Downloading autopage-0.5.0-py3-none-any.whl (29 kB)
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.9.0-py2.py3-none-any.whl (112 kB)
     |████████████████████████████████| 112 kB 67.2 MB/s
Requirement already satisfied: PrettyTable>=0.7.2 in /usr/local/lib/python3.7/dist
Collecting stevedore>=2.0.1
  Downloading stevedore-3.5.0-py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 4.4 MB/s
Collecting pyperclip>=1.6
  Downloading pyperclip-1.8.2.tar.gz (20 kB)
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/python3.7/dist-pac
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-
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, co
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)]))
  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)
clf_report = classification_report(Y_test, Y_pred, labels=[x for x in range(6)])
with open(f"./drive/MyDrive/SOLAR_CELL/ML_PROCESSED_DATA/outputs/classification_report
```

```
        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 = 5


    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: 0.49005 |  0:01:51s
epoch 46 | loss: 0.49429 |  0:01:54s
epoch 47 | loss: 0.48534 |  0:01:56s
epoch 48 | loss: 0.4829  |  0:01:59s
epoch 49 | loss: 0.47626 |  0:02:01s
epoch 50 | loss: 0.49432 |  0:02:04s
epoch 51 | loss: 0.48568 |  0:02:06s
epoch 52 | loss: 0.47402 |  0:02:08s
epoch 53 | loss: 0.46815 |  0:02:11s
epoch 54 | loss: 0.46226 |  0:02:13s
epoch 55 | loss: 0.46257 |  0:02:16s
epoch 56 | loss: 0.46696 |  0:02:18s
epoch 57 | loss: 0.46107 |  0:02:21s
epoch 58 | loss: 0.45831 |  0:02:23s
epoch 59 | loss: 0.45528 |  0:02:26s
epoch 60 | loss: 0.45837 |  0:02:28s
epoch 61 | loss: 0.45108 |  0:02:30s
epoch 62 | loss: 0.44977 |  0:02:33s
epoch 63 | loss: 0.46131 |  0:02:35s
epoch 64 | loss: 0.45492 |  0:02:38s
epoch 65 | loss: 0.44695 |  0:02:40s
epoch 66 | loss: 0.43846 |  0:02:43s
epoch 67 | loss: 0.43735 |  0:02:45s
epoch 68 | loss: 0.44016 |  0:02:47s
epoch 69 | loss: 0.4363  |  0:02:50s
epoch 70 | loss: 0.43019 |  0:02:52s
epoch 71 | loss: 0.42837 |  0:02:55s
epoch 72 | loss: 0.43008 |  0:02:57s
epoch 73 | loss: 0.43    |  0:02:59s
epoch 74 | loss: 0.4256  |  0:03:02s
epoch 75 | loss: 0.4242  |  0:03:04s
epoch 76 | loss: 0.42016 |  0:03:07s
epoch 77 | loss: 0.43779 |  0:03:09s
```

```
epoch 78 | loss: 0.41997 |  0:03:11s
epoch 79 | loss: 0.42372 |  0:03:14s
epoch 80 | loss: 0.41953 |  0:03:16s
epoch 81 | loss: 0.45579 |  0:03:19s
epoch 82 | loss: 0.43932 |  0:03:21s
epoch 83 | loss: 0.42653 |  0:03:23s
epoch 84 | loss: 0.42405 |  0:03:26s
epoch 85 | loss: 0.41615 |  0:03:28s
epoch 86 | loss: 0.40266 |  0:03:30s
epoch 87 | loss: 0.40685 |  0:03:33s
epoch 88 | loss: 0.40705 |  0:03:35s
epoch 89 | loss: 0.40919 |  0:03:37s
epoch 90 | loss: 0.40526 |  0:03:40s
epoch 91 | loss: 0.39636 |  0:03:42s
epoch 92 | loss: 0.41883 |  0:03:45s
epoch 93 | loss: 0.40969 |  0:03:47s
epoch 94 | loss: 0.39992 |  0:03:49s
epoch 95 | loss: 0.3956  |  0:03:52s
epoch 96 | loss: 0.39651 |  0:03:54s
epoch 97 | loss: 0.39868 |  0:03:56s
epoch 98 | loss: 0.3909  |  0:03:59s
epoch 99 | loss: 0.39294 |  0:04:01s
[++] Saving the model and parameters in corresponding directories
[++] Ended the training process for fold 5
```

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

Fold 2 has started running at 09:45 22-05-22

Fold 2 has completed sucessfully on 10:58 22-05-22

Fold 3 has started running at 18:40 22-05-22

Fold 3 has completed sucessfully on 22-05-22

Fold 4 completed sucessfully on 21:04 on 22-05-22

Fold 5 started at 18:21 on 23-05-22

✓ 1h 20m 4s completed at 7:41 PM