3_model_building_tuning_F13

November 18, 2022

0.1 # 3_model_building_LSTM_F13

Group , November 18, 2022 1. Eduardo Garcia 2. Nari Kim 3. Thi Anh Ba Dang 4. Vishnu Prabhakar 5. VS Chaitanya Madduri 6. Yumeng Zhang

Description: Applying 7 Algorithms on the daily data.

1. Dummy Predictor
2. Logistics Regresssion
3. Random Forest
4. XGBoost
5. LGBM
6. FNN
7. Tabnet

- We are builing a basic model using the columns and the target provided in the intial dataset.
- Used Feature Set 13 which are basic features and we have calculated the Lag , Ta metrics

0.1.1 Pre requisites:

1. And add the shortcut of the drive link : https://drive.google.com/drive/folders/1F8P3UlqSE6lFpHyBidVArd to your personal drive.

Files: crypto_data_hour_cleaned_v2.csv - Hourly Data

0.1.2 Output files:

Files:model files

- Observation : Overall we have build 6 models and optimised 3 models to see if we can get best results. Amoung all the models we see best results with Xgboost and LGBM

1 Load and transform data

[2]:
```
## Packages to be installed
!pip install ta
!pip install bayesian-optimization
!pip install pytorch-tabnet
!pip install tensorflow-addons
!pip install verstack
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting ta
  Downloading ta-0.10.2.tar.gz (25 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from ta) (1.21.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
(from ta) (1.3.5)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
packages (from pandas->ta) (2022.6)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas->ta) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.7.3->pandas->ta) (1.15.0)
Building wheels for collected packages: ta
  Building wheel for ta (setup.py) … done
  Created wheel for ta: filename=ta-0.10.2-py3-none-any.whl size=29104
sha256=1f34b0eaf75b52aa36763f4041ae531a95013a545a86ce62c0fc790f6c157dba
  Stored in directory: /root/.cache/pip/wheels/31/31/f1/f2ff471bbc5b84a4b973698c
eecdd453ae043971791adc3431
Successfully built ta
Installing collected packages: ta
Successfully installed ta-0.10.2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting bayesian-optimization
  Downloading bayesian_optimization-1.3.1-py3-none-any.whl (16 kB)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.7/dist-
packages (from bayesian-optimization) (1.21.6)
Requirement already satisfied: scikit-learn>=0.18.0 in
/usr/local/lib/python3.7/dist-packages (from bayesian-optimization) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-
packages (from bayesian-optimization) (1.7.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.18.0->bayesian-optimization) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.18.0->bayesian-
optimization) (3.1.0)
Installing collected packages: bayesian-optimization
Successfully installed bayesian-optimization-1.3.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pytorch-tabnet
  Downloading pytorch_tabnet-4.0-py3-none-any.whl (41 kB)
     |                 | 41 kB 537 kB/s
Requirement already satisfied: scipy>1.4 in /usr/local/lib/python3.7/dist-
packages (from pytorch-tabnet) (1.7.3)
Requirement already satisfied: numpy<2.0,>=1.17 in
```

```
/usr/local/lib/python3.7/dist-packages (from pytorch-tabnet) (1.21.6)
Requirement already satisfied: tqdm<5.0,>=4.36 in /usr/local/lib/python3.7/dist-
packages (from pytorch-tabnet) (4.64.1)
Requirement already satisfied: torch<2.0,>=1.2 in /usr/local/lib/python3.7/dist-
packages (from pytorch-tabnet) (1.12.1+cu113)
Requirement already satisfied: scikit_learn>0.21 in
/usr/local/lib/python3.7/dist-packages (from pytorch-tabnet) (1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit_learn>0.21->pytorch-tabnet)
(3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit_learn>0.21->pytorch-tabnet) (1.2.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.7/dist-packages (from torch<2.0,>=1.2->pytorch-tabnet)
(4.1.1)
Installing collected packages: pytorch-tabnet
Successfully installed pytorch-tabnet-4.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting tensorflow-addons
  Downloading tensorflow_addons-0.18.0-cp37-cp37m-manylinux_2_17_x86_64.manylinu
x2014_x86_64.whl (1.1 MB)
     |                 | 1.1 MB 4.8 MB/s
Requirement already satisfied: typeguard>=2.7 in
/usr/local/lib/python3.7/dist-packages (from tensorflow-addons) (2.7.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-
packages (from tensorflow-addons) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging->tensorflow-addons)
(3.0.9)
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.18.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting verstack
  Downloading verstack-3.2.5.tar.gz (9.6 MB)
     |                 | 9.6 MB 5.0 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-
packages (from verstack) (1.3.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from verstack) (1.21.6)
Requirement already satisfied: xgboost in /usr/local/lib/python3.7/dist-packages
(from verstack) (0.90)
Collecting scikit-learn==1.0.1
  Downloading
scikit_learn-1.0.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl
(23.2 MB)
     |                 | 23.2 MB 1.4 MB/s
```

```
Collecting lightgbm==3.3.0
  Downloading lightgbm-3.3.0-py3-none-manylinux1_x86_64.whl (2.0 MB)
     |                 | 2.0 MB 47.3 MB/s
Collecting optuna==2.10.0
  Downloading optuna-2.10.0-py3-none-any.whl (308 kB)
     |                 | 308 kB 69.4 MB/s
Collecting plotly==5.3.1
  Downloading plotly-5.3.1-py2.py3-none-any.whl (23.9 MB)
     |                 | 23.9 MB 1.3 MB/s
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.7/dist-packages (from verstack) (3.2.2)
Collecting python-dateutil==2.8.1
  Downloading python_dateutil-2.8.1-py2.py3-none-any.whl (227 kB)
     |                 | 227 kB 36.6 MB/s
Collecting holidays==0.11.3.1
  Downloading holidays-0.11.3.1-py3-none-any.whl (155 kB)
     |                 | 155 kB 66.7 MB/s
Requirement already satisfied: mlxtend in /usr/local/lib/python3.7/dist-
packages (from verstack) (0.14.0)
Collecting tensorflow==2.7.0
  Downloading https://us-python.pkg.dev/colab-wheels/public/tensorflow/tensorflo
w-2.7.0%2Bzzzcolab20220506150900-cp37-cp37m-linux_x86_64.whl (665.5 MB)
     |                 | 665.5 MB 19 kB/s
Collecting keras==2.7.0
  Downloading keras-2.7.0-py2.py3-none-any.whl (1.3 MB)
     |                 | 1.3 MB 63.6 MB/s
Collecting category_encoders==2.4.0
  Downloading category_encoders-2.4.0-py2.py3-none-any.whl (86 kB)
     |                 | 86 kB 5.5 MB/s
Requirement already satisfied: scipy>=1.0.0 in
/usr/local/lib/python3.7/dist-packages (from category_encoders==2.4.0->verstack)
(1.7.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.7/dist-
packages (from category_encoders==2.4.0->verstack) (0.5.3)
Requirement already satisfied: statsmodels>=0.9.0 in
/usr/local/lib/python3.7/dist-packages (from category_encoders==2.4.0->verstack)
(0.12.2)
Requirement already satisfied: convertdate>=2.3.0 in
/usr/local/lib/python3.7/dist-packages (from holidays==0.11.3.1->verstack)
(2.4.0)
Requirement already satisfied: hijri-converter in /usr/local/lib/python3.7/dist-
packages (from holidays==0.11.3.1->verstack) (2.2.4)
Requirement already satisfied: korean-lunar-calendar in
/usr/local/lib/python3.7/dist-packages (from holidays==0.11.3.1->verstack)
(0.3.1)
Requirement already satisfied: wheel in /usr/local/lib/python3.7/dist-packages
(from lightgbm==3.3.0->verstack) (0.38.3)
Collecting cmaes>=0.8.2
```

```
  Downloading cmaes-0.9.0-py3-none-any.whl (23 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from optuna==2.10.0->verstack) (4.64.1)
Collecting alembic
  Downloading alembic-1.8.1-py3-none-any.whl (209 kB)
     |                 | 209 kB 50.9 MB/s
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.7/dist-packages (from optuna==2.10.0->verstack) (21.3)
Collecting colorlog
  Downloading colorlog-6.7.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: sqlalchemy>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from optuna==2.10.0->verstack) (1.4.43)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.7/dist-packages
(from optuna==2.10.0->verstack) (6.0)
Collecting cliff
  Downloading cliff-3.10.1-py3-none-any.whl (81 kB)
     |                 | 81 kB 9.6 MB/s
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.7/dist-packages (from plotly==5.3.1->verstack) (8.1.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from plotly==5.3.1->verstack) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn==1.0.1->verstack)
(3.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn==1.0.1->verstack) (1.2.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(1.6.3)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(2.1.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==2.7.0->verstack) (1.14.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==2.7.0->verstack) (3.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(1.1.2)
Requirement already satisfied: gast<0.5.0,>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(0.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(4.1.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(1.50.0)
```

```
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(3.3.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==2.7.0->verstack) (3.19.6)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack) (1.12)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(0.27.0)
Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==2.7.0->verstack) (14.0.6)
Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow==2.7.0->verstack) (1.3.0)
Collecting tensorflow-estimator<2.8,~=2.7.0rc0
  Downloading tensorflow_estimator-2.7.0-py2.py3-none-any.whl (463 kB)
     |                 | 463 kB 69.4 MB/s
Requirement already satisfied: tensorboard~=2.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow==2.7.0->verstack)
(2.9.1)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in
/usr/local/lib/python3.7/dist-packages (from
convertdate>=2.3.0->holidays==0.11.3.1->verstack) (0.5.11)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-
packages (from h5py>=2.9.0->tensorflow==2.7.0->verstack) (1.5.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from
packaging>=20.0->optuna==2.10.0->verstack) (3.0.9)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-
packages (from pandas->verstack) (2022.6)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from
sqlalchemy>=1.1.0->optuna==2.10.0->verstack) (4.13.0)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.7/dist-packages (from
sqlalchemy>=1.1.0->optuna==2.10.0->verstack) (2.0.1)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (57.4.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (0.4.6)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-
packages (from tensorboard~=2.6->tensorflow==2.7.0->verstack) (1.0.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (1.8.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (2.14.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (0.6.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow==2.7.0->verstack) (2.23.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
packages (from tensorboard~=2.6->tensorflow==2.7.0->verstack) (3.4.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow==2.7.0->verstack) (5.2.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow==2.7.0->verstack) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow==2.7.0->verstack) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow==2.7.0->verstack) (1.3.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->sqlalchemy>=1.1.0->optuna==2.10.0->verstack)
(3.10.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow==2.7.0->verstack) (0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow==2.7.0->verstack) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow==2.7.0->verstack) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow==2.7.0->verstack) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow==2.7.0->verstack) (2022.9.24)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow==2.7.0->verstack) (3.2.2)
Collecting Mako
  Downloading Mako-1.2.4-py3-none-any.whl (78 kB)
```

```
     |                 | 78 kB 6.7 MB/s
Requirement already satisfied: importlib-resources in
/usr/local/lib/python3.7/dist-packages (from alembic->optuna==2.10.0->verstack)
(5.10.0)
Collecting pbr!=2.1.0,>=2.0.0
  Downloading pbr-5.11.0-py2.py3-none-any.whl (112 kB)
     |                 | 112 kB 67.0 MB/s
Collecting stevedore>=2.0.1
  Downloading stevedore-3.5.2-py3-none-any.whl (50 kB)
     |                 | 50 kB 6.1 MB/s
Collecting autopage>=0.4.0
  Downloading autopage-0.5.1-py3-none-any.whl (29 kB)
Requirement already satisfied: PrettyTable>=0.7.2 in
/usr/local/lib/python3.7/dist-packages (from cliff->optuna==2.10.0->verstack)
(3.5.0)
Collecting cmd2>=1.0.0
  Downloading cmd2-2.4.2-py3-none-any.whl (147 kB)
     |                 | 147 kB 54.3 MB/s
Requirement already satisfied: wcwidth>=0.1.7 in
/usr/local/lib/python3.7/dist-packages (from
cmd2>=1.0.0->cliff->optuna==2.10.0->verstack) (0.2.5)
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-
packages (from cmd2>=1.0.0->cliff->optuna==2.10.0->verstack) (22.1.0)
Requirement already satisfied: MarkupSafe>=0.9.2 in
/usr/local/lib/python3.7/dist-packages (from
Mako->alembic->optuna==2.10.0->verstack) (2.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->verstack) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->verstack) (0.11.0)
Building wheels for collected packages: verstack, pyperclip
  Building wheel for verstack (setup.py) … done
  Created wheel for verstack: filename=verstack-3.2.5-py3-none-any.whl
size=83395
sha256=82621d4069ca53242eff4410dceadb0d423a11515c68281b233991c6fb701704
  Stored in directory: /root/.cache/pip/wheels/43/4a/43/a38229cce0db60c2f7f124f2
9d15e4b6cbd5f28cdb7441d3ff
  Building wheel for pyperclip (setup.py) … done
  Created wheel for pyperclip: filename=pyperclip-1.8.2-py3-none-any.whl
size=11137
sha256=c58b1fac82e3a968c0369626de6020197a937745a9fb5e8d7a4e3dceaa20606e
  Stored in directory: /root/.cache/pip/wheels/9f/18/84/8f69f8b08169c7bae2dde6bd
7daf0c19fca8c8e500ee620a28
Successfully built verstack pyperclip
Installing collected packages: python-dateutil, pyperclip, pbr, stevedore, Mako,
cmd2, autopage, tensorflow-estimator, scikit-learn, keras, colorlog, cmaes,
```

```
cliff, alembic, tensorflow, plotly, optuna, lightgbm, holidays, category-
encoders, verstack
  Attempting uninstall: python-dateutil
    Found existing installation: python-dateutil 2.8.2
    Uninstalling python-dateutil-2.8.2:
      Successfully uninstalled python-dateutil-2.8.2
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.9.0
    Uninstalling tensorflow-estimator-2.9.0:
      Successfully uninstalled tensorflow-estimator-2.9.0
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.0.2
    Uninstalling scikit-learn-1.0.2:
      Successfully uninstalled scikit-learn-1.0.2
  Attempting uninstall: keras
    Found existing installation: keras 2.9.0
    Uninstalling keras-2.9.0:
      Successfully uninstalled keras-2.9.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.9.2
    Uninstalling tensorflow-2.9.2:
      Successfully uninstalled tensorflow-2.9.2
  Attempting uninstall: plotly
    Found existing installation: plotly 5.5.0
    Uninstalling plotly-5.5.0:
      Successfully uninstalled plotly-5.5.0
  Attempting uninstall: lightgbm
    Found existing installation: lightgbm 2.2.3
    Uninstalling lightgbm-2.2.3:
      Successfully uninstalled lightgbm-2.2.3
  Attempting uninstall: holidays
    Found existing installation: holidays 0.16
    Uninstalling holidays-0.16:
      Successfully uninstalled holidays-0.16
ERROR: pip's dependency resolver does not currently take into account all

the packages that are installed. This behaviour is the source of the following

dependency conflicts.

prophet 1.1.1 requires holidays>=0.14.2, but you have holidays 0.11.3.1 which is

incompatible.
Successfully installed Mako-1.2.4 alembic-1.8.1 autopage-0.5.1 category-
encoders-2.4.0 cliff-3.10.1 cmaes-0.9.0 cmd2-2.4.2 colorlog-6.7.0
holidays-0.11.3.1 keras-2.7.0 lightgbm-3.3.0 optuna-2.10.0 pbr-5.11.0
plotly-5.3.1 pyperclip-1.8.2 python-dateutil-2.8.1 scikit-learn-1.0.1
stevedore-3.5.2 tensorflow-2.7.0+zzzcolab20220506150900 tensorflow-
estimator-2.7.0 verstack-3.2.5
```

[6]:
```
## Import the packages
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import xgboost
import pickle
import tensorflow

# import tensorflow_addons as tfa
import torch

from pytorch_tabnet.tab_model import TabNetClassifier
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.models import Model
# from tensorflow.keras.optimizers import SGD, Adam
from sklearn.dummy import DummyClassifier
from ta import add_all_ta_features
from sklearn.model_selection import RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from xgboost.sklearn import XGBClassifier
from bayes_opt import BayesianOptimization
from sklearn.model_selection import cross_val_score
#picking models for prediction.
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
```

[7]:
```
from pip._internal.utils.misc import get_installed_distributions
import sys
#import numpy as np # imported to test whether numpy shows up, which it does!

def get_imported_packages():
    p = get_installed_distributions()
    p = {package.key:package.version for package in p}

    imported_modules = set(sys.modules.keys())

    imported_modules.remove('pip')

    modules = [(m, p[m]) for m in imported_modules if p.get(m, False)]
```

```
    return modules


def generate_requirements(filepath:str, modules):
    with open(filepath, 'w') as f:
        for module, version in modules:
            f.write(f"{module}=={version}")


generate_requirements('requirements.txt', get_imported_packages())
```

[ ]:
```
# file path
folder_path = '/content/drive/MyDrive/MADS_23_DL_final_project'
daily = pd.read_csv('crypto_data_daily_cleaned_v1.csv')
```

[ ]:
```
daily.head()
```

[ ]:

	Open Time	Open	High	Low	Close	Volume	train_test	Crypto
0	2013-04-01	93.155	105.90	93.155	104.750	11008.524	Train	BTC
1	2013-04-02	104.720	127.00	99.000	123.016	24187.398	Train	BTC
2	2013-04-03	123.001	146.88	101.511	125.500	31681.780	Train	BTC
3	2013-04-04	125.500	143.00	125.500	135.632	15035.206	Train	BTC
4	2013-04-05	136.000	145.00	135.119	142.990	11697.741	Train	BTC

2 Train / Test Split

[ ]:
```
# checking the Count of the records
daily['train_test'].value_counts(normalize=True)
```

[ ]:
```
Train    0.82546
Test     0.17454
Name: train_test, dtype: float64
```

[ ]:
```
df = daily.copy()
```

[ ]:
```
# train test split
train_df = df[df['train_test']=='Train']
test_df = df[df['train_test']=='Test']
```

[ ]:
```
train_df.head()
```

[ ]:

	Open Time	Open	High	Low	Close	Volume	train_test	Crypto
0	2013-04-01	93.155	105.90	93.155	104.750	11008.524	Train	BTC
1	2013-04-02	104.720	127.00	99.000	123.016	24187.398	Train	BTC
2	2013-04-03	123.001	146.88	101.511	125.500	31681.780	Train	BTC
3	2013-04-04	125.500	143.00	125.500	135.632	15035.206	Train	BTC

```
4  2013-04-05  136.000  145.00  135.119  142.990  11697.741      Train    BTC
```

```
[ ]: train_df.columns
```

```
[ ]: Index(['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume', 'train_test',
            'Crypto'],
           dtype='object')
```

3 Calculate percentage change

```
[ ]: def calculate_pct_change(df):
         '''
         The funnction calculate the pct Change for the all the records for one␣
      ↪period.
         input:
         df: The input dataFrame
         output:
         df_pct_change : The dataframe with percentage Calculated.
         '''
         coins = df.Crypto.unique()
         df_pct_change = pd.DataFrame()
         for coin in coins:
             x = df[df['Crypto']==coin]
             x['pct_change_1day'] = x['Close'].pct_change(1)
             df_pct_change = pd.concat([df_pct_change,x])
         return df_pct_change
```

```
[ ]: train_df = calculate_pct_change(train_df)
     test_df = calculate_pct_change(test_df)
```

```
[ ]: test_df.head()
```

```
[ ]:        Open Time      Open      High       Low     Close     Volume  \
     3098  2021-10-01  43828.89  48500.00  43287.44  48165.76  38375.517
     3099  2021-10-02  48185.61  48361.83  47438.00  47657.69  12310.011
     3100  2021-10-03  47649.00  49300.00  47119.87  48233.99  14411.104
     3101  2021-10-04  48233.99  49530.53  46895.80  49245.54  25695.213
     3102  2021-10-05  49244.13  51922.00  49057.18  51493.99  30764.491

          train_test Crypto  pct_change_1day
     3098       Test    BTC              NaN
     3099       Test    BTC        -0.010548
     3100       Test    BTC         0.012092
     3101       Test    BTC         0.020972
     3102       Test    BTC         0.045658
```

3.1 Generate new features

3.1.1 Lag features, moving average, exponential moving average and market cap

[ ]:
```
def create_market_volumn_features(df):
  '''
  We are calculating the market volumn and weighttage.
  input:
  df: input data set
  output:
  merged: the output dataset with Total_value and Value_weights are added.
  '''
  # calculate value of each cryto at certain time points
  df['Total_Value'] = df['Close']*df['Volume']
  # the sum of values at each time point
  sum_at_timepoints = df.groupby('Open Time').sum()['Total_Value']
  merged = df.merge(sum_at_timepoints, how='left',
                            on='Open Time', suffixes=('','_market'))
  merged['Value_Weight'] = merged['Total_Value']/merged['Total_Value_market']

  return merged
```

[ ]:
```
def create_shift_features(df, col = 'pct_change_1day',lags=10, freq='daily'):
  '''
    Creating the lag volatitlity for the data.
    input:
    df: the input dataframe
    col: the column on which the calculation should be made.
    lags: The periods for which the calculations should be made
    freq: Teh frequencey or time period
    output:
    df: the final dataframe withe lag columns added
  '''
  if freq=='daily':
    mul_fact = 1
    symbol = 'd'
  elif freq=='weekly':
    mul_fact = 7
    symbol = 'W'
  elif freq=='monthly':
    mul_fact = 31
    symbol = 'mon'
  else:
    # setting default to daily
    mul_fact = 1
    symbol = 'd'

  for iterator in range(1,lags+1):
```

```python
        df['{}_{}_lag'.format(iterator, symbol)] = df[col].
 ↪shift(periods=iterator*mul_fact)
        # df.loc[:,"Volatility_{}_{}".format(iterator, symbol)] = df[col].
 ↪rolling(iterator*mul_fact).std().shift(1)


    return df
```

[ ]:

[ ]:
```python
#list to collect all relevant lags

def create_analysis_colums(df):
    '''
        Create the additional features for the model building
        input:
        df: the input dataframe
        output:
        master_df: the final dataframe withe lag, ta columns added
    '''
    master_df = pd.DataFrame()
    crypto_coins = df['Crypto'].unique()

    for coin in crypto_coins:

        temp_df = df[df['Crypto']==coin]
        temp_df['pct_change_1day'] = temp_df['Close'].pct_change()

        # temp_df = create_shift_features(temp_df.copy(),col =␣
 ↪'pct_change_1day',lags=5, freq='weekly')
        temp_df = create_shift_features(temp_df.copy(),col =␣
 ↪'pct_change_1day',lags=2, freq='monthly')
        temp_df = create_shift_features(temp_df.copy(),col =␣
 ↪'pct_change_1day',lags=4, freq='weekly')
        temp_df = create_shift_features(temp_df.copy(),col =␣
 ↪'pct_change_1day',lags=8, freq='daily')
        temp_df = add_all_ta_features(temp_df.copy(), open="Open", high="High",␣
 ↪low="Low", close="Close", volume="Volume", fillna=True)
        if master_df.empty :
            master_df = temp_df
        else:
            master_df = pd.concat([master_df, temp_df])
    return master_df
```

[ ]:
```
# Creating the additonal columns for the model building
train_df =create_analysis_colums(train_df)
test_df =create_analysis_colums(test_df)
```

[ ]:
```
train_df.columns
```

[ ]:
```
Index(['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume', 'train_test',
       'Crypto', 'pct_change_1day', '1_mon_lag',
       …
       'momentum_ppo', 'momentum_ppo_signal', 'momentum_ppo_hist',
       'momentum_pvo', 'momentum_pvo_signal', 'momentum_pvo_hist',
       'momentum_kama', 'others_dr', 'others_dlr', 'others_cr'],
      dtype='object', length=109)
```

[ ]:
```
# adding the total_value ratio and market value columns
train_df =create_market_volumn_features(train_df.copy())
test_df =create_market_volumn_features(test_df.copy())
```

[ ]:
```
# printing the 50 columns
train_df.columns[:50]
```

[ ]:
```
Index(['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume', 'train_test',
       'Crypto', 'pct_change_1day', '1_mon_lag', '2_mon_lag', '1_W_lag',
       '2_W_lag', '3_W_lag', '4_W_lag', '1_d_lag', '2_d_lag', '3_d_lag',
       '4_d_lag', '5_d_lag', '6_d_lag', '7_d_lag', '8_d_lag', 'volume_adi',
       'volume_obv', 'volume_cmf', 'volume_fi', 'volume_em', 'volume_sma_em',
       'volume_vpt', 'volume_vwap', 'volume_mfi', 'volume_nvi',
       'volatility_bbm', 'volatility_bbh', 'volatility_bbl', 'volatility_bbw',
       'volatility_bbp', 'volatility_bbhi', 'volatility_bbli',
       'volatility_kcc', 'volatility_kch', 'volatility_kcl', 'volatility_kcw',
       'volatility_kcp', 'volatility_kchi', 'volatility_kcli',
       'volatility_dcl', 'volatility_dch', 'volatility_dcm'],
      dtype='object')
```

[ ]:
```
# Based on our previous model building activities we have selected specific␣
 ↪columns
feat_to_keep = ['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume',␣
 ↪'train_test',
       'Crypto',␣
 ↪'pct_change_1day','volatility_kcw','trend_cci','volume_adi','momentum_ppo_hist','momentum_s
             'volatility_dcw','volume_vpt','volatility_bbw','Total_Value',␣
 ↪'Total_Value_market', 'Value_Weight']
lag_cols = [col for col in train_df.columns if 'lag' in col]
#volatility_cols = [col for col in train_df.columns if 'Volatility' in col]
feat_to_keep.extend(lag_cols)
#feat_to_keep.extend(volatility_cols)
```

[ ]:
```
# length of the columns selected
len(feat_to_keep)
```

[ ]: 36

[ ]:
```
# slicing the tain and test columns
train_df = train_df[feat_to_keep]
test_df = test_df[feat_to_keep]
```

[ ]:
```
# length of the columns selected
len(train_df.columns)
```

[ ]: 36

[ ]:
```
# extracting the importatant features
impo_feat =␣
↪['volatility_kcw','trend_cci','volume_adi','momentum_ppo_hist','momentum_stoch','volatility
         'volatility_dcw','volume_vpt','volatility_bbw']
for feat in impo_feat:
  train_df[feat] = train_df[feat].shift(1)
  test_df[feat] = test_df[feat].shift(1)
```

3.2 Extract year, month, day, hour and weekday from time stamp

3.2.1 Encoding of ordinals

[ ]:
```
def encode_cyclicals(df_x):
    '''
    The function converts the date features encoded in the Sine and cosines.
    Input :
    df_x : Input data frame to be processed
    Output :
    df_x : processed dataframe.
    '''
    df_x['month_sin'] = np.sin(2*np.pi*df_x.month/12)
    df_x['month_cos'] = np.cos(2*np.pi*df_x.month/12)
    df_x.drop('month', axis=1, inplace=True)

    df_x['day_sin'] = np.sin(2*np.pi*df_x.day/31)
    df_x['day_cos'] = np.cos(2*np.pi*df_x.day/31)
    df_x.drop('day', axis=1, inplace=True)

    df_x['dayofweek_sin'] = np.sin(2*np.pi*df_x.weekday/7)
    df_x['dayofweek_cos'] = np.cos(2*np.pi*df_x.weekday/7)
    df_x.drop('weekday', axis=1, inplace=True)

    df_x['hour_sin'] = np.sin(2*np.pi*df_x.hour/24)
    df_x['hour_cos'] = np.cos(2*np.pi*df_x.hour/24)
```

```
    df_x.drop('hour', axis=1, inplace=True)

    df_x['hour_sin'] = np.sin(2*np.pi*df_x.minute/60)
    df_x['hour_cos'] = np.cos(2*np.pi*df_x.minute/60)
    df_x.drop('minute', axis=1, inplace=True)



    return df_x
```

[ ]:
```
def date_values_extraction(new_df):
    '''
  The function to split the date columns.
  Input :
  new_df : Input data frame to be processed
  Output :
  df : processed dataframe.
  '''
  df = new_df.copy()
  df['year'] = pd.DatetimeIndex(df['Open Time']).year
  df['month'] = pd.DatetimeIndex(df['Open Time']).month
  df['day'] = pd.DatetimeIndex(df['Open Time']).day
  df['weekday'] = pd.DatetimeIndex(df['Open Time']).dayofweek


  df['Open Time'] = pd.to_datetime(df['Open Time'])
  df['minute'] = df['Open Time'].dt.minute
  df['hour'] = df['Open Time'].dt.hour
  df = encode_cyclicals(df.copy())
  return df
```

[ ]:
```
# Extract the date featues
train_df = date_values_extraction(train_df)
test_df = date_values_extraction(test_df)
```

3.3 One hot coding the coins

[ ]:
```
# Applying one hot encoding on Crypto Coin
def crypto_one_hot_encoding(df):
  '''
  The funtion which converts the crypto into individual columns.
  Input :
  df : Input data frame to be processed
  Output :
  df : processed dataframe.
  '''
  y_dummies = pd.get_dummies(df['Crypto'], prefix='Crypto', drop_first= False)
```

```
    df = pd.concat([df, y_dummies], axis=1)
    df.drop(['Crypto'], axis=1, inplace=True)
    # creating a additional column if the model is used for new coin.
    df['other_crypto'] =0
    return df
```

[ ]:
```
# The train and test where one hot encoded
train_df = crypto_one_hot_encoding(train_df)
test_df = crypto_one_hot_encoding(test_df)
```

[ ]:
```
train_df['pct_change_1day'].describe()
```

[ ]:
```
count    17115.000000
mean         0.005533
std          0.162083
min         -0.564847
25%         -0.025641
50%          0.000000
75%          0.028824
max         19.058824
Name: pct_change_1day, dtype: float64
```

[ ]:
```
test_df['pct_change_1day'].describe()
```

[ ]:
```
count    3611.000000
mean       -0.001452
std         0.046840
min        -0.204696
25%        -0.026274
50%         0.000000
75%         0.024091
max         0.344702
Name: pct_change_1day, dtype: float64
```

4 Defining the Target Variable.

We devide the data into 3 classes, one is with return below 0, one is above 0 but below market rate of return, one is above market rate of return. The market rate of return we use here is annual return of S&P 500 index in 2021.

[ ]:
```
def create_target(df):
    '''
    The funtion which extracts the target columns from pct_change_1day
    Input :
    df : Input data frame to be processed
    Output :
    df : processed dataframe.
```

```
    '''
    market_RoR = 26.89
    market_RoR_1d = market_RoR/365
    df['Target'] = np.where(df['pct_change_1day']>0, 1,0)
    df['Target'] = np.where(df['pct_change_1day']>market_RoR_1d, 2,1)
    df['Target'][df['Target']==1] = np.
 ↪where(df['pct_change_1day'][df['Target']==1]>=0, 1,0)
    return df
```

[ ]:
```
# Applying the target columns
train_df = create_target(train_df)
test_df = create_target(test_df)
```

[ ]:
```
train_df['Target'].value_counts(normalize=True)
```

[ ]:
```
0    0.460088
1    0.449343
2    0.090569
Name: Target, dtype: float64
```

[ ]:
```
test_df['Target'].value_counts(normalize=True)
```

[ ]:
```
0    0.491025
1    0.464513
2    0.044463
Name: Target, dtype: float64
```

[ ]:
```
test_df.drop(['pct_change_1day'], axis=1, inplace=True) # droppping the column␣
 ↪as we already extracted the target
train_df.drop(['pct_change_1day'], axis=1, inplace=True) # droppping the column␣
 ↪as we already extracted the target
```

[ ]:
```
train_df.shape
```

[ ]:
```
(17125, 55)
```

[ ]:
```
test_df.shape
```

[ ]:
```
(3621, 55)
```

[ ]:
```
target = train_df['Target']
test_target = test_df['Target']
```

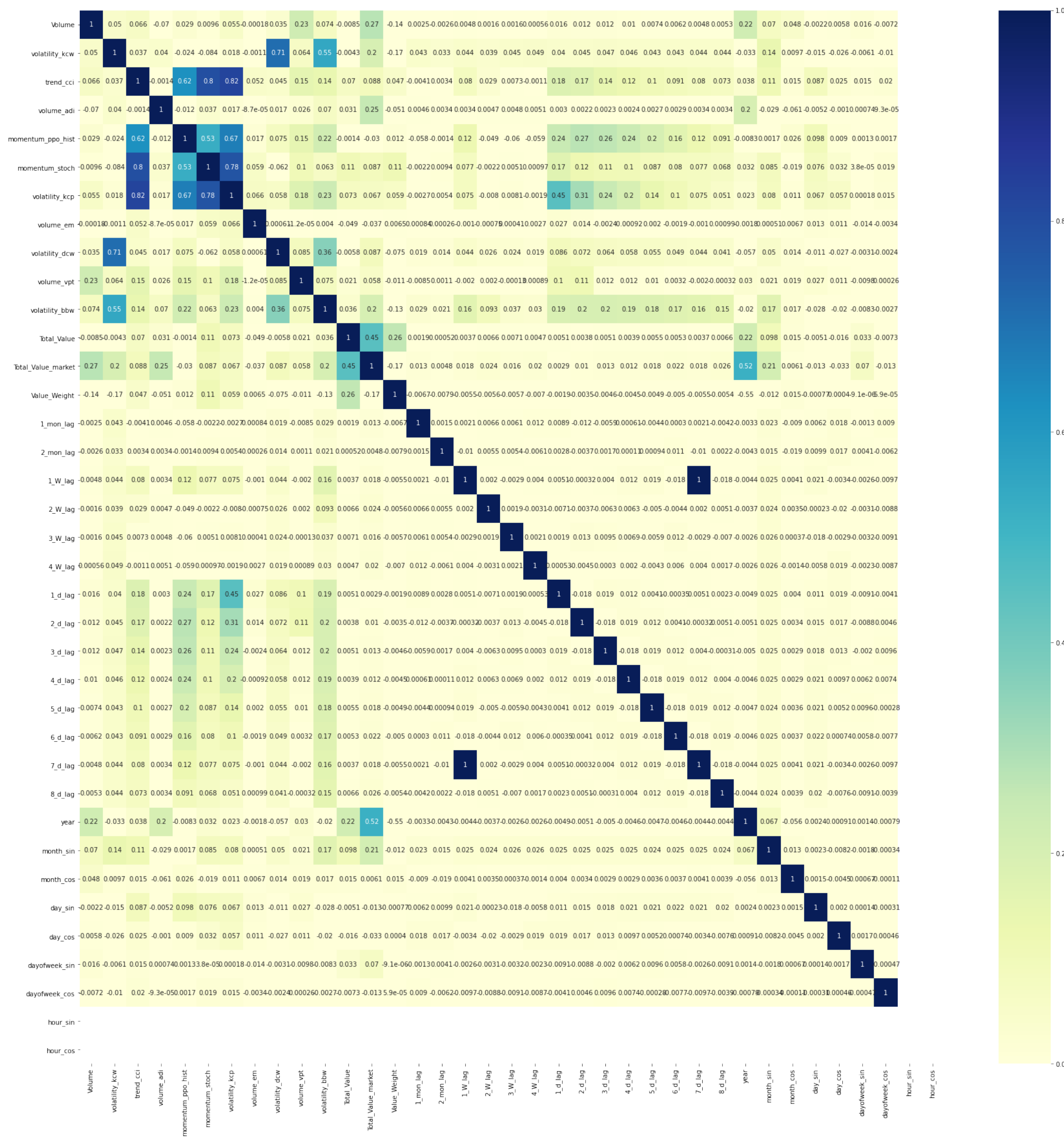
4.0.1 Drop columns

[ ]:
```
train_df.drop(['Target','Open Time','train_test',],axis=1,inplace=True)
```

[ ]:
```
test_df.drop(['Target','Open Time','train_test',],axis=1,inplace=True)
```

[ ]:
```
plt.figure(figsize=(30,30))

corr = train_df.drop(['Open','High', 'Low','Close','Crypto_ADA',
       'Crypto_BTC', 'Crypto_ETC', 'Crypto_ETH', 'Crypto_LINK', 'Crypto_LTC',
       'Crypto_TRX', 'Crypto_XLM', 'Crypto_XMR', 'Crypto_XRP',␣
 ↪'other_crypto'],axis=1).corr()

sns.heatmap(corr,vmin=0, vmax=1, annot=True, cmap="YlGnBu")
```

[ ]: <AxesSubplot:>



- The correlation matrix shows that columns are not much correlated each other.

[ ]:
```
# dropping the listof the columns
# drop_columns = []
drop_columns = ['Open','Close']


if drop_columns:
  norm_train_df = train_df.drop(drop_columns,axis=1)
  norm_test_df = test_df.drop(drop_columns,axis=1)
else:
  norm_train_df = train_df
  norm_test_df = test_df
```

[ ]:
```
norm_train_df.columns
```

[ ]:
```
Index(['High', 'Low', 'Volume', 'volatility_kcw', 'trend_cci', 'volume_adi',
       'momentum_ppo_hist', 'momentum_stoch', 'volatility_kcp', 'volume_em',
       'volatility_dcw', 'volume_vpt', 'volatility_bbw', 'Total_Value',
       'Total_Value_market', 'Value_Weight', '1_mon_lag', '2_mon_lag',
       '1_W_lag', '2_W_lag', '3_W_lag', '4_W_lag', '1_d_lag', '2_d_lag',
       '3_d_lag', '4_d_lag', '5_d_lag', '6_d_lag', '7_d_lag', '8_d_lag',
       'year', 'month_sin', 'month_cos', 'day_sin', 'day_cos', 'dayofweek_sin',
       'dayofweek_cos', 'hour_sin', 'hour_cos', 'Crypto_ADA', 'Crypto_BTC',
       'Crypto_ETC', 'Crypto_ETH', 'Crypto_LINK', 'Crypto_LTC', 'Crypto_TRX',
       'Crypto_XLM', 'Crypto_XMR', 'Crypto_XRP', 'other_crypto'],
      dtype='object')
```

[ ]:
```
norm_train_df.shape
```

[ ]:
```
(17125, 50)
```

[ ]:
```
target.shape
```

[ ]:
```
(17125,)
```

[ ]:
```
def generate_model_report(y_actual, y_predicted, metric_type):
    '''
    This function we are calculating the main metrics
    input:
    y_actual: the actual values
    y_predicted : The predicted values
    metric_type : which type of metric
    '''
    print("==================Printing the {} metrics===================".
 ↪format(metric_type))
    if metric_type=='micro':
      print("Accuracy = " , round(accuracy_score(y_actual, y_predicted),3))
```

```
    print("Precision = " ,round(precision_score(y_actual, y_predicted,␣
↪average=metric_type),3))
    print("Recall = " ,round(recall_score(y_actual, y_predicted,␣
↪average=metric_type),3))
    print("F1 Score = " ,round(f1_score(y_actual, y_predicted,␣
↪average=metric_type),3))
    print("=====================================")
```

5 Model Building

5.1 Dummy classifier

The baseline Model which others model should be beating.

[ ]:
```
# Fitting the Dummt classifier

dummy_model = DummyClassifier(strategy='prior')

dummy_model.fit(train_df.fillna(0), target)
```

[ ]:
```
DummyClassifier()
```

[ ]:
```
# predicting the values
dummy_pred = dummy_model.predict(test_df)
```

[ ]:
```
# Evaluation metrics
generate_model_report(test_target, dummy_pred, 'micro')
generate_model_report(test_target, dummy_pred, 'macro')
generate_model_report(test_target, dummy_pred, 'weighted')


print('\nClassification Report\n')
print(classification_report(test_target, dummy_pred))
```

```
==================Printing the micro metrics===================
Accuracy =  0.491
Precision =  0.491
Recall =  0.491
F1 Score =  0.491
=====================================
==================Printing the macro metrics===================
Precision =  0.164
Recall =  0.333
F1 Score =  0.22
=====================================
==================Printing the weighted metrics===================
Precision =  0.241
Recall =  0.491
```

```
F1 Score =  0.323
=====================================

Classification Report

              precision    recall  f1-score   support

           0       0.49      1.00      0.66      1778
           1       0.00      0.00      0.00      1682
           2       0.00      0.00      0.00       161

    accuracy                           0.49      3621
   macro avg       0.16      0.33      0.22      3621
weighted avg       0.24      0.49      0.32      3621
```

[ ]:
```
# Saving the file
import pickle
# temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/LGBM_hourly'
with open( 'dummy_f13_final.pkl', 'wb') as f:  # Python 3: open(..., 'wb')
    pickle.dump(dummy_model, f)
```

5.2 Logistic Regression

[ ]:
```
# Building the Logistic Regression Model

logreg = LogisticRegression(multi_class='multinomial', solver='lbfgs')         ␣
↪                   # Set Large C value for low regularization to prevent␣
↪overfitting
logreg.fit(train_df.fillna(0), target)
```

[ ]:
```
LogisticRegression(multi_class='multinomial')
```

[ ]:
```
# Predicting teh
logreg_pred = logreg.predict(test_df.fillna(0))
```

[ ]:
```
generate_model_report(test_target, logreg_pred, 'micro')
generate_model_report(test_target, logreg_pred, 'macro')
generate_model_report(test_target, logreg_pred, 'weighted')

from sklearn.metrics import classification_report
print('\nClassification Report\n')
print(classification_report(test_target, logreg_pred))
```

```
==================Printing the micro metrics===================
Accuracy =  0.493
Precision =  0.493
```

```
Recall =  0.493
F1 Score =  0.493
=====================================
==================Printing the macro metrics===================
Precision =  0.353
Recall =  0.335
F1 Score =  0.233
=====================================
==================Printing the weighted metrics===================
Precision =  0.505
Recall =  0.493
F1 Score =  0.341
=====================================

Classification Report

              precision    recall  f1-score   support

           0       0.49      0.98      0.66      1778
           1       0.56      0.02      0.04      1682
           2       0.00      0.00      0.00       161

    accuracy                           0.49      3621
   macro avg       0.35      0.33      0.23      3621
weighted avg       0.50      0.49      0.34      3621
```

5.3 XGBoost - Untuned

[ ]:
```
%timeit
# Fitting the untuned xgclassifier
xgb_clf = XGBClassifier(num_class = 2,
                        objective="multi:softprob",
                        n_estimators = 1000,
                        max_depth = 30,
                        eval_metric ='mlogloss', n_jobs=-1)
xgb_clf.fit(norm_train_df, target)
```

[ ]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='mlogloss', gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=30, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=1000, n_jobs=-1,
              num_class=2, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=None, subsample=1, tree_method='exact',
```

```
                 validate_parameters=1, …)
```

[ ]:
```python
xgb_clf
```

[ ]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='mlogloss', gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=30, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=1000, n_jobs=-1,
              num_class=2, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, reg_lambda=1,
              scale_pos_weight=None, subsample=1, tree_method='exact',
              validate_parameters=1, …)
```

[ ]:
```python
# Predicting the values
predicted_values = xgb_clf.predict(norm_test_df)
```

[ ]:
```python
# Evaluation metrics
generate_model_report(test_target, predicted_values, 'micro')
generate_model_report(test_target, predicted_values, 'macro')
generate_model_report(test_target, predicted_values, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.5
Precision =  0.5
Recall =  0.5
F1 Score =  0.5
=====================================
==================Printing the macro metrics===================
Precision =  0.396
Recall =  0.399
F1 Score =  0.393
=====================================
==================Printing the weighted metrics===================
Precision =  0.506
Recall =  0.5
F1 Score =  0.497
=====================================
```

[ ]:
```python
print('\nClassification Report\n')
print(classification_report(test_target, predicted_values))
```

```

Classification Report

              precision    recall  f1-score   support
```

```
           0       0.54      0.42      0.48      1778
           1       0.50      0.61      0.55      1682
           2       0.14      0.16      0.15       161

    accuracy                           0.50      3621
   macro avg       0.40      0.40      0.39      3621
weighted avg       0.51      0.50      0.50      3621
```

[ ]:
```
# Saving the file

temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/LGBM_hourly'
with open( 'XGBoost_f13_final.pkl', 'wb') as f:  # Python 3: open(..., 'wb')
    pickle.dump(xgb_clf, f)
```

[ ]:
```

```

5.4 XGBoost tuning with Bayesian optimization

[ ]:
```
def xgbc_cv( max_depth,
                n_estimators,
                gamma,
                learning_rate,
                min_child_weight,

                 subsample):



    estimator_function = XGBClassifier(max_depth=int(max_depth),
                                           learning_rate= learning_rate,
                                           n_estimators= int(n_estimators),
                                           min_child_weight =␣
↪int(min_child_weight),
                                           nthread = -1,
                                           subsample =  max(subsample, 0),
                                           objective="multi:softprob",
                                           eval_metric = 'mlogloss',
                                           num_class = 2,
                                           scoring='f1',
                                           seed = 32)
    # Fit the estimator

    estimator_function.fit(norm_train_df.fillna(0), target)

    # calculate out-of-the-box roc_score using validation set 1
```

```python
    predicted_values = estimator_function.predict(norm_test_df.fillna(0))
    score = f1_score(test_target, predicted_values, average='weighted')


    # return the mean validation score to be maximized
    return score


def bayesOpt(train_x, train_y):
    lgbBO = BayesianOptimization(xgbc_cv, {
                                              "max_depth": (25,30),
                                              "n_estimators": (600,1000) ,
                                              "gamma": (0.03,0.05),
                                              "learning_rate": (0.04,0.09 ),
                                              "min_child_weight": (5,20),
                                              # "colsample_bytree":(0.4,0.8),
                                              "subsample": (0.50,0.85)
                                              }
                                 )


    lgbBO.maximize(init_points=5, n_iter=30)

#     print(lgbBO.res['max'])
    print(lgbBO.max)


bayesOpt(norm_train_df.fillna(0), target)
```

```
|   iter    |  target   |   gamma   | learni… | max_depth | min_ch… |
n_esti… | subsample |
--------------------------------------------------------------------------------
-----------------
[16:22:10] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 1         | 0.4862    | 0.03422   | 0.08214
| 25.59     | 11.82     | 616.8     | 0.7609
|
```

```
[16:24:16] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 2         | 0.4901    | 0.04502   | 0.07011

 | 26.72     | 6.628     | 631.0     |
0.7776    |
[16:26:50] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 3         | 0.4972    | 0.03127   | 0.04498

 | 29.4      | 11.72     | 686.8     |
0.7095    |
[16:29:42] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 4         | 0.4892    | 0.03314   | 0.06276
| 29.75     | 8.247     | 750.4     | 0.8352
|
[16:33:11] WARNING: C:/Users/Administrator/workspace/xgboost-
```

```
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 5         | 0.486     | 0.03415   | 0.06178
| 28.5      | 17.08     | 860.1     | 0.6472
|
[16:36:02] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 6         | 0.494     | 0.04503   | 0.08116
| 27.62     | 6.331     | 630.9     | 0.7636
|
[16:38:37] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 7         | 0.4951    | 0.04831   | 0.06202
| 26.95     | 5.39      | 845.3     | 0.8457
|
[16:41:55] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.
```

```
  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 8         | 0.4874    | 0.0399    | 0.05872
| 27.13     | 18.58     | 871.2     | 0.8134
|
[16:45:15] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 9         | 0.492     | 0.04666   | 0.0884
| 26.08     | 11.55     | 997.5     | 0.8295
|
[16:48:29] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 10        | 0.4881    | 0.04044   | 0.05047
| 29.6      | 8.288     | 778.2     | 0.8094
|
[16:51:53] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
```

```
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 11        | 0.4867    | 0.04216   | 0.06782
| 29.57     | 19.55     | 782.6     | 0.8378
|
[16:54:37] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 12        | 0.4891    | 0.03932   | 0.06761
| 25.89     | 5.024     | 852.4     | 0.6084
|
[16:56:37] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 13        | 0.4889    | 0.04309   | 0.06445
| 29.48     | 11.22     | 686.7     | 0.6452
|
[16:58:18] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.
```

```
| 14        | 0.4866    | 0.03937   | 0.06116
| 25.39     | 9.467     | 623.8     | 0.8171
|
[17:00:20] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 15        | 0.482     | 0.04285   | 0.07545
| 27.15     | 13.31     | 821.8     | 0.6563
|
[17:02:50] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 16        | 0.4894    | 0.04647   | 0.07957
| 27.57     | 5.832     | 868.4     | 0.7801
|
[17:05:29] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 17        | 0.4888    | 0.03528   | 0.0715
```

```
| 29.38     | 13.8      | 937.7     | 0.8089
|
[17:08:40] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 18        | 0.487     | 0.03343   | 0.08868
| 29.2      | 12.64     | 777.1     | 0.8339
|
[17:10:48] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 19        | 0.4809    | 0.0418    | 0.0769
| 29.0      | 15.18     | 822.2     | 0.5286
|
[17:12:23] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 20        | 0.4757    | 0.04211   | 0.08836
| 29.43     | 11.65     | 687.1     | 0.5939
|
[17:14:29] WARNING: C:/Users/Administrator/workspace/xgboost-
```

```
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 21        | 0.4883    | 0.03734   | 0.07968
| 26.58     | 17.99     | 744.1     | 0.6516
|
[17:16:46] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 22        | 0.4885    | 0.04956   | 0.04758
| 29.93     | 9.908     | 734.9     | 0.6196
|
[17:19:31] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 23        | 0.4848    | 0.03149   | 0.08573
| 25.2      | 19.09     | 794.3     | 0.5296
|
[17:21:08] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.
```

```
  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 24        | 0.4884    | 0.0391    | 0.04025
| 26.18     | 12.37     | 716.3     | 0.7432
|
[17:23:33] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 25        | 0.4917    | 0.0472    | 0.08877
| 25.35     | 15.33     | 604.0     | 0.5446
|
[17:25:06] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 26        | 0.4926    | 0.03763   | 0.04227
| 26.35     | 13.05     | 603.3     | 0.5615
|
[17:27:10] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
```

```
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 27        | 0.4899    | 0.03195   | 0.06928
| 25.57     | 17.49     | 639.7     | 0.7907
|
[17:29:29] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 28        | 0.4865    | 0.04464   | 0.05685
| 28.22     | 19.67     | 774.9     | 0.8011
|
[17:32:39] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 29        | 0.4776    | 0.03237   | 0.08219
| 28.23     | 18.52     | 608.9     | 0.6754
|
[17:34:26] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.
```

```
| 30        | 0.4929    | 0.04517   | 0.05115
| 27.94     | 13.46     | 709.4     | 0.806
|
[17:37:08] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 31        | 0.4831    | 0.04625   | 0.0843
| 26.4      | 12.3      | 733.7     | 0.5804
|
[17:38:56] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 32        | 0.4842    | 0.03994   | 0.06386
| 27.52     | 19.31     | 759.8     | 0.7816
|
[17:40:58] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 33        | 0.4801    | 0.04962   | 0.08702
```

```
| 25.51     | 19.42     | 819.3     | 0.6651
|
[17:43:03] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 34        | 0.4891    | 0.04654   | 0.06936
| 27.57     | 5.118     | 777.0     | 0.624
|
[17:45:53] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "scoring" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.


| 35        | 0.4905    | 0.03478   | 0.06467
| 28.9      | 5.686     | 657.1     | 0.6216
|
================================================================================
=================
{'target': 0.49717263911048104, 'params': {'gamma': 0.03127013296857564,
'learning_rate': 0.04498314951689869, 'max_depth': 29.401153112346318,
'min_child_weight': 11.719617320022135, 'n_estimators': 686.8341805804602,
'subsample': 0.7094565386859188}}
```

5.5 RandomizedSearchCV - XGBoost

[ ]:
```
# Importing RandomizedSearchCV
```

[ ]:
```
# EG Fitting 3 folds for each of 100 candidates, totalling 300 fits
xg_random = RandomizedSearchCV(estimator = xgb_clf,
                               param_distributions = param_grid,
                               n_iter = 10,
```

```
                                cv = 3,
                                verbose=1 ,
                                scoring='f1',
                                n_jobs=-1,
                                random_state=42)
```

[ ]:
```
xg_random.fit(norm_train_df, target)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

[ ]:
```
RandomizedSearchCV(cv=3,
                   estimator=XGBClassifier(eval_metric='mlogloss', max_depth=30,
                                           n_jobs=-1, num_class=2,
                                           objective='multi:softprob'),
                   n_jobs=-1,
                   param_distributions={'colsample_bytree': [0.4, 0.8],
                                        'gamma': [0.03, 0.05],
                                        'learning_rate': [0.02, 0.09],
                                        'min_child_weight': [5, 10],
                                        'n_estimators': [300, 400, 500],
                                        'subsample': [0.5, 0.85]},
                   random_state=42, scoring='f1', verbose=1)
```

[ ]:
```
xg_random.cv_results_
```

[ ]:
```
{'mean_fit_time': array([284.74294734, 533.94794067, 324.38395802, 273.51564415,
        217.38379757, 293.17546924, 320.41527597, 107.98551663,
        143.0566748 , 188.16923523]),
 'std_fit_time': array([16.54543366, 39.83647681, 22.32838181, 19.46776648,
15.59290301,
        21.21693124, 28.53303815,  7.15326628,  9.65456451, 14.22315131]),
 'mean_score_time': array([ 4.39842884, 17.06368883,  5.13770858,  2.45175536,
8.86828566,
         3.52253485,  2.5757939 ,  1.60393675,  3.2699577 ,  7.45017727]),
 'std_score_time': array([0.29605027, 1.46149761, 0.1760782 , 0.10329311,
1.60875665,
        0.98636956, 0.13675698, 0.13713717, 0.52527974, 0.8564662 ]),
 'param_subsample': masked_array(data=[0.5, 0.85, 0.85, 0.5, 0.85, 0.85, 0.85,
0.5, 0.5, 0.5],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'param_n_estimators': masked_array(data=[400, 500, 300, 500, 400, 300, 400,
300, 300, 500],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
```

```
             dtype=object),
 'param_min_child_weight': masked_array(data=[10, 5, 5, 10, 10, 10, 10, 10, 5,
10],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'param_learning_rate': masked_array(data=[0.02, 0.02, 0.02, 0.09, 0.02, 0.02,
0.09, 0.09, 0.02,
                    0.02],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'param_gamma': masked_array(data=[0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.03,
0.05, 0.03,
                    0.03],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'param_colsample_bytree': masked_array(data=[0.8, 0.8, 0.8, 0.8, 0.4, 0.8, 0.8,
0.4, 0.4, 0.4],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'params': [{'subsample': 0.5,
   'n_estimators': 400,
   'min_child_weight': 10,
   'learning_rate': 0.02,
   'gamma': 0.05,
   'colsample_bytree': 0.8},
  {'subsample': 0.85,
   'n_estimators': 500,
   'min_child_weight': 5,
   'learning_rate': 0.02,
   'gamma': 0.05,
   'colsample_bytree': 0.8},
  {'subsample': 0.85,
   'n_estimators': 300,
   'min_child_weight': 5,
   'learning_rate': 0.02,
   'gamma': 0.05,
   'colsample_bytree': 0.8},
  {'subsample': 0.5,
   'n_estimators': 500,
```

```
  'min_child_weight': 10,
  'learning_rate': 0.09,
  'gamma': 0.05,
  'colsample_bytree': 0.8},
 {'subsample': 0.85,
  'n_estimators': 400,
  'min_child_weight': 10,
  'learning_rate': 0.02,
  'gamma': 0.05,
  'colsample_bytree': 0.4},
 {'subsample': 0.85,
  'n_estimators': 300,
  'min_child_weight': 10,
  'learning_rate': 0.02,
  'gamma': 0.05,
  'colsample_bytree': 0.8},
 {'subsample': 0.85,
  'n_estimators': 400,
  'min_child_weight': 10,
  'learning_rate': 0.09,
  'gamma': 0.03,
  'colsample_bytree': 0.8},
 {'subsample': 0.5,
  'n_estimators': 300,
  'min_child_weight': 10,
  'learning_rate': 0.09,
  'gamma': 0.05,
  'colsample_bytree': 0.4},
 {'subsample': 0.5,
  'n_estimators': 300,
  'min_child_weight': 5,
  'learning_rate': 0.02,
  'gamma': 0.03,
  'colsample_bytree': 0.4},
 {'subsample': 0.5,
  'n_estimators': 500,
  'min_child_weight': 10,
  'learning_rate': 0.02,
  'gamma': 0.03,
  'colsample_bytree': 0.4}],
'split0_test_score': array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]),
'split1_test_score': array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]),
'split2_test_score': array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]),
'mean_test_score': array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]),
'std_test_score': array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]),
'rank_test_score': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10],
dtype=int32)}
```

```
[ ]: xg_random
```

```
[ ]: RandomizedSearchCV(cv=3,
                        estimator=XGBClassifier(eval_metric='mlogloss', max_depth=30,
                                                n_jobs=-1, num_class=2,
                                                objective='multi:softprob'),
                        n_jobs=-1,
                        param_distributions={'colsample_bytree': [0.4, 0.8],
                                             'gamma': [0.03, 0.05],
                                             'learning_rate': [0.02, 0.09],
                                             'min_child_weight': [5, 10],
                                             'n_estimators': [300, 400, 500],
                                             'subsample': [0.5, 0.85]},
                        random_state=42, scoring='f1', verbose=1)
```

```
[ ]: # Picking teh best estimator
final_model = xg_random.best_estimator_
```

```
[ ]: # Saving the file

temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
 ↪model_files/svc_hourly'
with open(temp_file_path + '/xgb_clf_f13.pkl', 'wb') as f:  # Python 3: open(...
 ↪, 'wb')
    pickle.dump(final_model, f)
```

```
[ ]: # Predicting the values
predicted_values = final_model.predict(norm_test_df)
```

```
[ ]: # calculating the metrics
generate_model_report(test_target, predicted_values, 'micro')
generate_model_report(test_target, predicted_values, 'macro')
generate_model_report(test_target, predicted_values, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.511
Precision =  0.511
Recall =  0.511
F1 Score =  0.511
=====================================
==================Printing the macro metrics===================
Precision =  0.394
Recall =  0.373
F1 Score =  0.373
=====================================
==================Printing the weighted metrics===================
Precision =  0.503
Recall =  0.511
```

```
F1 Score =  0.501
=====================================
```

[ ]:
```python
# Best params
param_grid =  {'gamma': 0.03127013296857564, 'learning_rate': 0.
↪04498314951689869, 'max_depth': 29.401153112346318, 'min_child_weight': 11.
↪719617320022135, 'n_estimators': 686.8341805804602, 'subsample': 0.
↪7094565386859188}
```

[ ]:
```python
param_grid
```

[ ]:
```
{'gamma': 0.03127013296857564,
 'learning_rate': 0.04498314951689869,
 'max_depth': 29.401153112346318,
 'min_child_weight': 11.719617320022135,
 'n_estimators': 686.8341805804602,
 'subsample': 0.7094565386859188}
```

[ ]:
```python
%timeit

xgb_clf = XGBClassifier(num_class = 2,

                        learning_rate =param_grid['learning_rate'],
                        n_estimators = int(param_grid['n_estimators']),
                        gamma= param_grid['gamma'],
                        max_depth = int(param_grid['max_depth']),
                        subsample= param_grid['subsample'],
                        objective="multi:softprob",
                        eval_metric = 'mlogloss',
                        n_jobs=-1,num_boost_round=15)


xgb_clf.fit(norm_train_df.fillna(0), target)
```

```
[20:06:53] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.0/src/learner.cc:576:
Parameters: { "num_boost_round" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.
```

[ ]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='mlogloss', gamma=0.03127013296857564, gpu_id=-1,
              importance_type=None, interaction_constraints='',
              learning_rate=0.04498314951689869, max_delta_step=0, max_depth=29,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=686, n_jobs=-1, num_boost_round=15, num_class=2,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=0.7094565386859188, tree_method='exact', …)
```

[ ]:
```
xgb_clf
```

[ ]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              eval_metric='mlogloss', gamma=0.03127013296857564, gpu_id=-1,
              importance_type=None, interaction_constraints='',
              learning_rate=0.04498314951689869, max_delta_step=0, max_depth=29,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=686, n_jobs=-1, num_boost_round=15, num_class=2,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=0.7094565386859188, tree_method='exact', …)
```

[ ]:
```
predicted_values = xgb_clf.predict(norm_test_df)
```

[ ]:
```
generate_model_report(test_target, predicted_values, 'micro')
generate_model_report(test_target, predicted_values, 'macro')
generate_model_report(test_target, predicted_values, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.502
Precision =  0.502
Recall =  0.502
F1 Score =  0.502
=====================================
==================Printing the macro metrics===================
Precision =  0.381
Recall =  0.379
F1 Score =  0.374
=====================================
==================Printing the weighted metrics===================
Precision =  0.507
Recall =  0.502
F1 Score =  0.496
=====================================
```

[ ]:
```python
# Saving the file
import pickle
# temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/svc_hourly'
with open('xgb_clf_f13_tuned.pkl', 'wb') as f:  # Python 3: open(..., 'wb')
    pickle.dump(final_model, f)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_33188/2713381438.py in <module>
      2 import pickle
      3 temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/svc_hourly'
----> 4 with open(temp_file_path + '/xgb_clf_f13_tuned.pkl', 'wb') as f:  #␣
↪Python 3: open(…, 'wb')
      5     pickle.dump(final_model, f)

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/MyDrive/
↪MADS_23_DL_final_project/data/model_files/svc_hourly/xgb_clf_f13_tuned.pkl'
```

[ ]:
```python
print('\nClassification Report\n')
print(classification_report(test_target, predicted_values))
```

```
Classification Report

              precision    recall  f1-score   support

           0       0.54      0.42      0.48      1778
           1       0.50      0.61      0.55      1682
           2       0.14      0.16      0.15       161

    accuracy                           0.50      3621
   macro avg       0.40      0.40      0.39      3621
weighted avg       0.51      0.50      0.50      3621
```

[ ]:
```python
xgb_clf
```

5.6 Fully Connected Neural Network

[ ]:
```python
# USE KERAS FUNKTIONAl API!



# Parameters
############
```

```
hidden_size_1 = 1024
hidden_size_2 = 2048
hidden_size_3 = 1024
number_of_classes =3

# Model
#######

inputs = Input(shape=(norm_train_df.fillna(0).shape[1]))

# Hidden layer
##############

hidden_output = Dense(hidden_size_1, activation='relu')(inputs)
hidden_output = Dropout(0.5)(hidden_output)
hidden_output_2 = Dense(hidden_size_2, activation='relu')(hidden_output)
hidden_output_2 = Dropout(0.3)(hidden_output_2)
hidden_output_3 = Dense(hidden_size_3, activation='relu')(hidden_output_2)

# Softmax
#########


predictions = Dense(3, activation='softmax')(hidden_output_3)

# Whole model
##############
# Nothing more is left, than to instantiate the model
# Please ensure input and output is right!

model = Model(inputs=inputs, outputs=predictions)

# Optimization
##############
# For now, we stick to this.
optimizer = Adam(lr=0.01)


# Compilation and teaching
#########################

model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy', # use this cross entropy␣
 ↪variant
                                                       # since the input is not␣
 ↪one-hot encoded
```

```
             metrics=[ 'accuracy']) #We measure and print accuracy during␣
↪training
```

[ ]:
```
model.summary()
```

```
Model: "model_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 54)]              0

 dense_8 (Dense)             (None, 1024)              56320

 dropout_4 (Dropout)         (None, 1024)              0

 dense_9 (Dense)             (None, 2048)              2099200

 dropout_5 (Dropout)         (None, 2048)              0

 dense_10 (Dense)            (None, 1024)              2098176

 dense_11 (Dense)            (None, 3)                 3075

=================================================================
Total params: 4,256,771
Trainable params: 4,256,771
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
# fitting the model
model.fit(x=norm_train_df.fillna(0),
          y=target,
          validation_data=(norm_test_df.fillna(0), test_target),
          epochs=30,
          batch_size=32)
```

```
Epoch 1/30
536/536 [==============================] - 19s 34ms/step - loss: 85875304.0000 -
accuracy: 0.4516 - val_loss: 0.8601 - val_accuracy: 0.4910
Epoch 2/30
536/536 [==============================] - 18s 34ms/step - loss: 5603.3916 -
accuracy: 0.4548 - val_loss: 0.8589 - val_accuracy: 0.4910
Epoch 3/30
536/536 [==============================] - 19s 36ms/step - loss: 20654.1816 -
accuracy: 0.4558 - val_loss: 0.8585 - val_accuracy: 0.4910
Epoch 4/30
536/536 [==============================] - 18s 34ms/step - loss: 914.0923 -
accuracy: 0.4566 - val_loss: 0.8556 - val_accuracy: 0.4645
```

```
Epoch 5/30
536/536 [==============================] - 19s 35ms/step - loss: 3122.2930 -
accuracy: 0.4604 - val_loss: 0.8577 - val_accuracy: 0.4910
Epoch 6/30
536/536 [==============================] - 19s 35ms/step - loss: 459.7038 -
accuracy: 0.4575 - val_loss: 0.8591 - val_accuracy: 0.4645
Epoch 7/30
536/536 [==============================] - 19s 35ms/step - loss: 219.5765 -
accuracy: 0.4569 - val_loss: 0.8639 - val_accuracy: 0.4645
Epoch 8/30
536/536 [==============================] - 18s 34ms/step - loss: 0.9347 -
accuracy: 0.4566 - val_loss: 0.8572 - val_accuracy: 0.4645
Epoch 9/30
536/536 [==============================] - 19s 35ms/step - loss: 0.9349 -
accuracy: 0.4517 - val_loss: 0.8625 - val_accuracy: 0.4645
Epoch 10/30
536/536 [==============================] - 18s 34ms/step - loss: 0.9346 -
accuracy: 0.4546 - val_loss: 0.8597 - val_accuracy: 0.4645
Epoch 11/30
536/536 [==============================] - 19s 35ms/step - loss: 66.3475 -
accuracy: 0.4558 - val_loss: 0.8637 - val_accuracy: 0.4910
Epoch 12/30
536/536 [==============================] - 18s 34ms/step - loss: 0.9347 -
accuracy: 0.4525 - val_loss: 0.8586 - val_accuracy: 0.4910
Epoch 13/30
536/536 [==============================] - 18s 34ms/step - loss: 17270.8906 -
accuracy: 0.4464 - val_loss: 0.8577 - val_accuracy: 0.4910
Epoch 14/30
536/536 [==============================] - 18s 34ms/step - loss: 13979.8789 -
accuracy: 0.4554 - val_loss: 0.8612 - val_accuracy: 0.4910
Epoch 15/30
536/536 [==============================] - 19s 35ms/step - loss: 100.4182 -
accuracy: 0.4586 - val_loss: 0.8556 - val_accuracy: 0.4910
Epoch 16/30
536/536 [==============================] - 18s 34ms/step - loss: 619.7780 -
accuracy: 0.4547 - val_loss: 0.8619 - val_accuracy: 0.4910
Epoch 17/30
536/536 [==============================] - 18s 33ms/step - loss: 0.9349 -
accuracy: 0.4497 - val_loss: 0.8563 - val_accuracy: 0.4645
Epoch 18/30
536/536 [==============================] - 18s 34ms/step - loss: 0.9347 -
accuracy: 0.4566 - val_loss: 0.8587 - val_accuracy: 0.4645
Epoch 19/30
536/536 [==============================] - 18s 33ms/step - loss: 0.9346 -
accuracy: 0.4629 - val_loss: 0.8637 - val_accuracy: 0.4645
Epoch 20/30
536/536 [==============================] - 19s 35ms/step - loss: 0.9351 -
accuracy: 0.4508 - val_loss: 0.8597 - val_accuracy: 0.4910
```

```
Epoch 21/30
536/536 [==============================] - 18s 34ms/step - loss: 0.9346 -
accuracy: 0.4603 - val_loss: 0.8609 - val_accuracy: 0.4645
Epoch 22/30
536/536 [==============================] - 19s 35ms/step - loss: 0.9348 -
accuracy: 0.4535 - val_loss: 0.8626 - val_accuracy: 0.4910
Epoch 23/30
536/536 [==============================] - 18s 33ms/step - loss: 60779.2734 -
accuracy: 0.4591 - val_loss: 0.8596 - val_accuracy: 0.4645
Epoch 24/30
536/536 [==============================] - 19s 35ms/step - loss: 398.0602 -
accuracy: 0.4549 - val_loss: 0.8583 - val_accuracy: 0.4645
Epoch 25/30
536/536 [==============================] - 18s 34ms/step - loss: 22489.8516 -
accuracy: 0.4555 - val_loss: 0.8607 - val_accuracy: 0.4645
Epoch 26/30
536/536 [==============================] - 18s 33ms/step - loss: 20542.8906 -
accuracy: 0.4533 - val_loss: 0.8591 - val_accuracy: 0.4910
Epoch 27/30
536/536 [==============================] - 19s 35ms/step - loss: 1063.1647 -
accuracy: 0.4500 - val_loss: 0.8550 - val_accuracy: 0.4910
Epoch 28/30
536/536 [==============================] - 18s 33ms/step - loss: 10535.9346 -
accuracy: 0.4562 - val_loss: 0.8562 - val_accuracy: 0.4645
Epoch 29/30
536/536 [==============================] - 19s 35ms/step - loss: 0.9348 -
accuracy: 0.4564 - val_loss: 0.8589 - val_accuracy: 0.4645
Epoch 30/30
536/536 [==============================] - 18s 33ms/step - loss: 0.9348 -
accuracy: 0.4572 - val_loss: 0.8617 - val_accuracy: 0.4910
```

[ ]: <keras.callbacks.History at 0x7fdcb7d34450>

[ ]:
```
# Predicting the test data
nn_pred=model.predict(norm_test_df.fillna(0))
```

```
114/114 [==============================] - 8s 68ms/step
```

[ ]:
```
# Extracting the argmax
df = pd.DataFrame(nn_pred)
nn_pred = df.idxmax(axis=1)
```

[ ]:
```
# Evaluation metrics
generate_model_report(test_target, nn_pred, 'micro')
generate_model_report(test_target, nn_pred, 'macro')
generate_model_report(test_target, nn_pred, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.4645
```

```
Precision =  0.4645
Recall =  0.4645
F1 Score =  0.4645
=====================================
==================Printing the macro metrics===================
Precision =  0.1548
Recall =  0.3333
F1 Score =  0.2115
=====================================
==================Printing the weighted metrics===================
Precision =  0.2158
Recall =  0.4645
F1 Score =  0.2947
=====================================
```

[ ]:
```
print('\nClassification Report\n')
print(classification_report(test_target, nn_pred))
```

5.7 Tabnet

[ ]:
```
# define the model and fit
clf1_nopreproc = TabNetClassifier(optimizer_fn=torch.optim.Adam,
                       optimizer_params=dict(lr=2e-2),
                       scheduler_params={"step_size":10, # how to use learning␣
 ↪rate scheduler
                                         "gamma":0.9},
                       scheduler_fn=torch.optim.lr_scheduler.StepLR,
                       mask_type='entmax' # "sparsemax"
                      )

# fit the model
clf1_nopreproc.fit(
    norm_train_df.fillna(0).values, target.values,
    eval_set=[(norm_train_df.fillna(0).values, target.values), (norm_test_df.
 ↪fillna(0).values, test_target.values)],
    eval_name=['train', 'valid'],
    eval_metric=['balanced_accuracy'],
    max_epochs=150 , patience=120,
    batch_size=64, virtual_batch_size=128,
    num_workers=0,
    weights=1,
    drop_last=False
)
```

```
epoch 0  | loss: 1.08729 | train_balanced_accuracy: 0.42932 |
valid_balanced_accuracy: 0.39624 |  0:00:06s
epoch 1  | loss: 1.03804 | train_balanced_accuracy: 0.45699 |
valid_balanced_accuracy: 0.41154 |  0:00:15s
```

```
epoch 2  | loss: 1.03205 | train_balanced_accuracy: 0.44434 |
valid_balanced_accuracy: 0.37765 |  0:00:23s
epoch 3  | loss: 1.0262  | train_balanced_accuracy: 0.46195 |
valid_balanced_accuracy: 0.38644 |  0:00:29s
epoch 4  | loss: 1.0118  | train_balanced_accuracy: 0.44718 |
valid_balanced_accuracy: 0.35146 |  0:00:36s
epoch 5  | loss: 1.00969 | train_balanced_accuracy: 0.46104 |
valid_balanced_accuracy: 0.38105 |  0:00:43s
epoch 6  | loss: 1.00283 | train_balanced_accuracy: 0.47166 |
valid_balanced_accuracy: 0.4123  |  0:00:50s
epoch 7  | loss: 1.00314 | train_balanced_accuracy: 0.48128 |
valid_balanced_accuracy: 0.40423 |  0:00:56s
epoch 8  | loss: 0.99158 | train_balanced_accuracy: 0.47601 |
valid_balanced_accuracy: 0.39889 |  0:01:02s
epoch 9  | loss: 0.99393 | train_balanced_accuracy: 0.47143 |
valid_balanced_accuracy: 0.35349 |  0:01:09s
epoch 10 | loss: 0.98888 | train_balanced_accuracy: 0.50008 |
valid_balanced_accuracy: 0.36751 |  0:01:16s
epoch 11 | loss: 0.96833 | train_balanced_accuracy: 0.49872 |
valid_balanced_accuracy: 0.37467 |  0:01:23s
epoch 12 | loss: 0.9655  | train_balanced_accuracy: 0.50002 |
valid_balanced_accuracy: 0.3877  |  0:01:29s
epoch 13 | loss: 0.95158 | train_balanced_accuracy: 0.49681 |
valid_balanced_accuracy: 0.37479 |  0:01:35s
epoch 14 | loss: 0.95198 | train_balanced_accuracy: 0.50588 |
valid_balanced_accuracy: 0.35936 |  0:01:42s
epoch 15 | loss: 0.94276 | train_balanced_accuracy: 0.51632 |
valid_balanced_accuracy: 0.37477 |  0:01:48s
epoch 16 | loss: 0.91723 | train_balanced_accuracy: 0.49726 |
valid_balanced_accuracy: 0.35505 |  0:01:55s
epoch 17 | loss: 0.92767 | train_balanced_accuracy: 0.52077 |
valid_balanced_accuracy: 0.39867 |  0:02:01s
epoch 18 | loss: 0.91834 | train_balanced_accuracy: 0.52945 |
valid_balanced_accuracy: 0.37378 |  0:02:08s
epoch 19 | loss: 0.90961 | train_balanced_accuracy: 0.53995 |
valid_balanced_accuracy: 0.35771 |  0:02:14s
epoch 20 | loss: 0.90027 | train_balanced_accuracy: 0.52485 |
valid_balanced_accuracy: 0.37378 |  0:02:21s
epoch 21 | loss: 0.89831 | train_balanced_accuracy: 0.54238 |
valid_balanced_accuracy: 0.37217 |  0:02:27s
epoch 22 | loss: 0.88976 | train_balanced_accuracy: 0.54312 |
valid_balanced_accuracy: 0.36684 |  0:02:34s
epoch 23 | loss: 0.8859  | train_balanced_accuracy: 0.53463 |
valid_balanced_accuracy: 0.37275 |  0:02:40s
epoch 24 | loss: 0.88835 | train_balanced_accuracy: 0.56075 |
valid_balanced_accuracy: 0.3878  |  0:02:47s
epoch 25 | loss: 0.87744 | train_balanced_accuracy: 0.55285 |
valid_balanced_accuracy: 0.37537 |  0:02:53s
```

```
epoch 26 | loss: 0.87619 | train_balanced_accuracy: 0.55923 |
valid_balanced_accuracy: 0.35302 |  0:02:59s
epoch 27 | loss: 0.8626  | train_balanced_accuracy: 0.55779 |
valid_balanced_accuracy: 0.36577 |  0:03:06s
epoch 28 | loss: 0.85214 | train_balanced_accuracy: 0.55575 |
valid_balanced_accuracy: 0.37775 |  0:03:12s
epoch 29 | loss: 0.85189 | train_balanced_accuracy: 0.56313 |
valid_balanced_accuracy: 0.36688 |  0:03:19s
epoch 30 | loss: 0.85508 | train_balanced_accuracy: 0.54049 |
valid_balanced_accuracy: 0.36918 |  0:03:25s
epoch 31 | loss: 0.85084 | train_balanced_accuracy: 0.58154 |
valid_balanced_accuracy: 0.37334 |  0:03:32s
epoch 32 | loss: 0.84567 | train_balanced_accuracy: 0.55808 |
valid_balanced_accuracy: 0.36088 |  0:03:38s
epoch 33 | loss: 0.84306 | train_balanced_accuracy: 0.55913 |
valid_balanced_accuracy: 0.35683 |  0:03:44s
epoch 34 | loss: 0.83348 | train_balanced_accuracy: 0.57997 |
valid_balanced_accuracy: 0.35898 |  0:03:51s
epoch 35 | loss: 0.83438 | train_balanced_accuracy: 0.57298 |
valid_balanced_accuracy: 0.374   |  0:03:58s
epoch 36 | loss: 0.83231 | train_balanced_accuracy: 0.56677 |
valid_balanced_accuracy: 0.38235 |  0:04:04s
epoch 37 | loss: 0.82796 | train_balanced_accuracy: 0.58484 |
valid_balanced_accuracy: 0.36922 |  0:04:10s
epoch 38 | loss: 0.82502 | train_balanced_accuracy: 0.58426 |
valid_balanced_accuracy: 0.36818 |  0:04:17s
epoch 39 | loss: 0.81531 | train_balanced_accuracy: 0.57663 |
valid_balanced_accuracy: 0.36517 |  0:04:23s
epoch 40 | loss: 0.82914 | train_balanced_accuracy: 0.58323 |
valid_balanced_accuracy: 0.37707 |  0:04:29s
epoch 41 | loss: 0.81902 | train_balanced_accuracy: 0.5873  |
valid_balanced_accuracy: 0.37172 |  0:04:36s
epoch 42 | loss: 0.81443 | train_balanced_accuracy: 0.58255 |
valid_balanced_accuracy: 0.38053 |  0:04:43s
epoch 43 | loss: 0.80589 | train_balanced_accuracy: 0.59259 |
valid_balanced_accuracy: 0.40054 |  0:04:49s
epoch 44 | loss: 0.80995 | train_balanced_accuracy: 0.59656 |
valid_balanced_accuracy: 0.3901  |  0:04:56s
epoch 45 | loss: 0.79916 | train_balanced_accuracy: 0.59152 |
valid_balanced_accuracy: 0.39101 |  0:05:02s
epoch 46 | loss: 0.7951  | train_balanced_accuracy: 0.59592 |
valid_balanced_accuracy: 0.37794 |  0:05:09s
epoch 47 | loss: 0.78743 | train_balanced_accuracy: 0.57897 |
valid_balanced_accuracy: 0.39122 |  0:05:15s
epoch 48 | loss: 0.79812 | train_balanced_accuracy: 0.60352 |
valid_balanced_accuracy: 0.37721 |  0:05:22s
epoch 49 | loss: 0.79636 | train_balanced_accuracy: 0.61286 |
valid_balanced_accuracy: 0.37466 |  0:05:28s
```

```
epoch 50 | loss: 0.79574 | train_balanced_accuracy: 0.59492 |
valid_balanced_accuracy: 0.39868 |  0:05:34s
epoch 51 | loss: 0.7922  | train_balanced_accuracy: 0.61761 |
valid_balanced_accuracy: 0.387   |  0:05:41s
epoch 52 | loss: 0.78775 | train_balanced_accuracy: 0.60871 |
valid_balanced_accuracy: 0.39676 |  0:05:47s
epoch 53 | loss: 0.78033 | train_balanced_accuracy: 0.61305 |
valid_balanced_accuracy: 0.39993 |  0:05:53s
epoch 54 | loss: 0.79279 | train_balanced_accuracy: 0.59919 |
valid_balanced_accuracy: 0.39267 |  0:06:00s
epoch 55 | loss: 0.79476 | train_balanced_accuracy: 0.58554 |
valid_balanced_accuracy: 0.40512 |  0:06:06s
epoch 56 | loss: 0.79529 | train_balanced_accuracy: 0.57335 |
valid_balanced_accuracy: 0.36169 |  0:06:12s
epoch 57 | loss: 0.78514 | train_balanced_accuracy: 0.59644 |
valid_balanced_accuracy: 0.39014 |  0:06:19s
epoch 58 | loss: 0.77798 | train_balanced_accuracy: 0.61522 |
valid_balanced_accuracy: 0.39945 |  0:06:25s
epoch 59 | loss: 0.78279 | train_balanced_accuracy: 0.60367 |
valid_balanced_accuracy: 0.37908 |  0:06:31s
epoch 60 | loss: 0.7816  | train_balanced_accuracy: 0.61073 |
valid_balanced_accuracy: 0.37613 |  0:06:38s
epoch 61 | loss: 0.77906 | train_balanced_accuracy: 0.60585 |
valid_balanced_accuracy: 0.39357 |  0:06:45s
epoch 62 | loss: 0.77346 | train_balanced_accuracy: 0.6147  |
valid_balanced_accuracy: 0.39014 |  0:06:51s
epoch 63 | loss: 0.76619 | train_balanced_accuracy: 0.61722 |
valid_balanced_accuracy: 0.379   |  0:06:58s
epoch 64 | loss: 0.76873 | train_balanced_accuracy: 0.60593 |
valid_balanced_accuracy: 0.38366 |  0:07:04s
epoch 65 | loss: 0.76239 | train_balanced_accuracy: 0.6151  |
valid_balanced_accuracy: 0.38979 |  0:07:10s
epoch 66 | loss: 0.76005 | train_balanced_accuracy: 0.59914 |
valid_balanced_accuracy: 0.38029 |  0:07:17s
epoch 67 | loss: 0.76569 | train_balanced_accuracy: 0.62932 |
valid_balanced_accuracy: 0.39355 |  0:07:24s
epoch 68 | loss: 0.76006 | train_balanced_accuracy: 0.6195  |
valid_balanced_accuracy: 0.39069 |  0:07:30s
epoch 69 | loss: 0.76618 | train_balanced_accuracy: 0.61029 |
valid_balanced_accuracy: 0.40029 |  0:07:36s
epoch 70 | loss: 0.75801 | train_balanced_accuracy: 0.60774 |
valid_balanced_accuracy: 0.36723 |  0:07:43s
epoch 71 | loss: 0.76232 | train_balanced_accuracy: 0.61565 |
valid_balanced_accuracy: 0.38205 |  0:07:49s
epoch 72 | loss: 0.75112 | train_balanced_accuracy: 0.61485 |
valid_balanced_accuracy: 0.38412 |  0:07:56s
epoch 73 | loss: 0.75863 | train_balanced_accuracy: 0.59589 |
valid_balanced_accuracy: 0.37215 |  0:08:03s
```

```
epoch 74 | loss: 0.75479 | train_balanced_accuracy: 0.60204 |
valid_balanced_accuracy: 0.39199 |  0:08:09s
epoch 75 | loss: 0.76071 | train_balanced_accuracy: 0.62331 |
valid_balanced_accuracy: 0.39126 |  0:08:15s
epoch 76 | loss: 0.75277 | train_balanced_accuracy: 0.62375 |
valid_balanced_accuracy: 0.39737 |  0:08:22s
epoch 77 | loss: 0.75332 | train_balanced_accuracy: 0.59101 |
valid_balanced_accuracy: 0.38389 |  0:08:28s
epoch 78 | loss: 0.7528  | train_balanced_accuracy: 0.60685 |
valid_balanced_accuracy: 0.3768  |  0:08:35s
epoch 79 | loss: 0.75348 | train_balanced_accuracy: 0.6036  |
valid_balanced_accuracy: 0.37572 |  0:08:41s
epoch 80 | loss: 0.74556 | train_balanced_accuracy: 0.61106 |
valid_balanced_accuracy: 0.36929 |  0:08:48s
epoch 81 | loss: 0.75218 | train_balanced_accuracy: 0.59921 |
valid_balanced_accuracy: 0.38006 |  0:08:54s
epoch 82 | loss: 0.7429  | train_balanced_accuracy: 0.59807 |
valid_balanced_accuracy: 0.37381 |  0:09:01s
epoch 83 | loss: 0.74328 | train_balanced_accuracy: 0.62759 |
valid_balanced_accuracy: 0.38178 |  0:09:07s
epoch 84 | loss: 0.7386  | train_balanced_accuracy: 0.61717 |
valid_balanced_accuracy: 0.3739  |  0:09:13s
epoch 85 | loss: 0.75087 | train_balanced_accuracy: 0.61999 |
valid_balanced_accuracy: 0.39013 |  0:09:20s
epoch 86 | loss: 0.75104 | train_balanced_accuracy: 0.62011 |
valid_balanced_accuracy: 0.38935 |  0:09:27s
epoch 87 | loss: 0.75049 | train_balanced_accuracy: 0.60756 |
valid_balanced_accuracy: 0.37154 |  0:09:33s
epoch 88 | loss: 0.73765 | train_balanced_accuracy: 0.60142 |
valid_balanced_accuracy: 0.37245 |  0:09:40s
epoch 89 | loss: 0.7395  | train_balanced_accuracy: 0.59368 |
valid_balanced_accuracy: 0.37856 |  0:09:46s
epoch 90 | loss: 0.7403  | train_balanced_accuracy: 0.60837 |
valid_balanced_accuracy: 0.38233 |  0:09:52s
epoch 91 | loss: 0.74567 | train_balanced_accuracy: 0.61556 |
valid_balanced_accuracy: 0.37167 |  0:09:59s
epoch 92 | loss: 0.74558 | train_balanced_accuracy: 0.61199 |
valid_balanced_accuracy: 0.38302 |  0:10:06s
epoch 93 | loss: 0.73227 | train_balanced_accuracy: 0.60764 |
valid_balanced_accuracy: 0.37717 |  0:10:12s
epoch 94 | loss: 0.73832 | train_balanced_accuracy: 0.60615 |
valid_balanced_accuracy: 0.38735 |  0:10:19s
epoch 95 | loss: 0.73214 | train_balanced_accuracy: 0.63771 |
valid_balanced_accuracy: 0.38581 |  0:10:25s
epoch 96 | loss: 0.72661 | train_balanced_accuracy: 0.63612 |
valid_balanced_accuracy: 0.38305 |  0:10:31s
epoch 97 | loss: 0.73308 | train_balanced_accuracy: 0.64282 |
valid_balanced_accuracy: 0.38917 |  0:10:38s
```

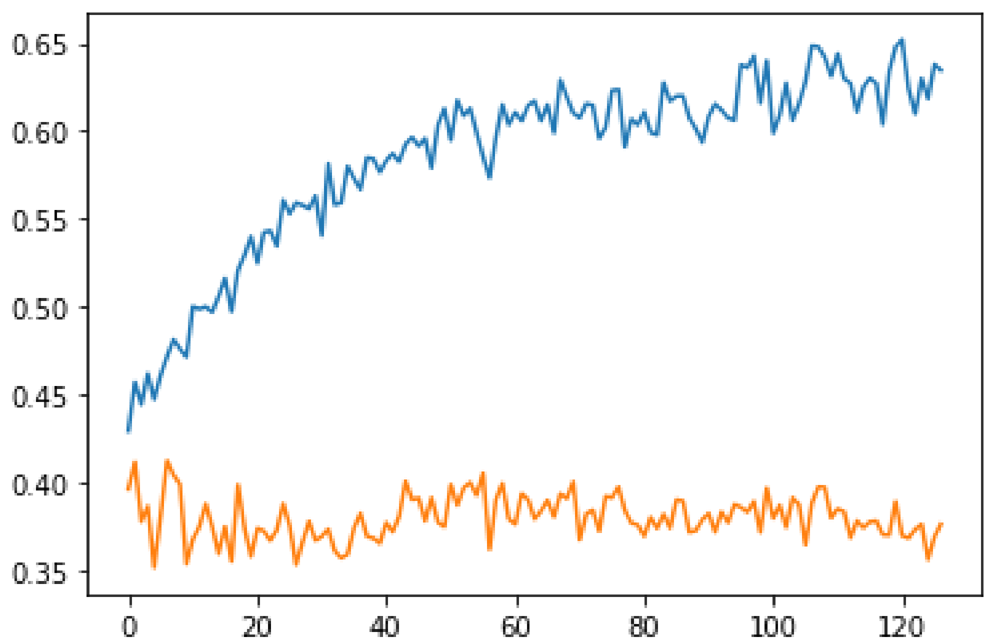
```
epoch 98 | loss: 0.73029 | train_balanced_accuracy: 0.61611 |
valid_balanced_accuracy: 0.37151 |  0:10:45s
epoch 99 | loss: 0.73767 | train_balanced_accuracy: 0.64017 |
valid_balanced_accuracy: 0.39671 |  0:10:51s
epoch 100| loss: 0.74229 | train_balanced_accuracy: 0.59897 |
valid_balanced_accuracy: 0.37954 |  0:10:57s
epoch 101| loss: 0.73277 | train_balanced_accuracy: 0.60913 |
valid_balanced_accuracy: 0.38695 |  0:11:04s
epoch 102| loss: 0.73808 | train_balanced_accuracy: 0.62746 |
valid_balanced_accuracy: 0.37431 |  0:11:10s
epoch 103| loss: 0.73009 | train_balanced_accuracy: 0.60618 |
valid_balanced_accuracy: 0.39121 |  0:11:17s
epoch 104| loss: 0.72669 | train_balanced_accuracy: 0.61565 |
valid_balanced_accuracy: 0.38722 |  0:11:23s
epoch 105| loss: 0.72675 | train_balanced_accuracy: 0.62828 |
valid_balanced_accuracy: 0.36445 |  0:11:30s
epoch 106| loss: 0.72745 | train_balanced_accuracy: 0.64856 |
valid_balanced_accuracy: 0.38814 |  0:11:36s
epoch 107| loss: 0.72924 | train_balanced_accuracy: 0.64804 |
valid_balanced_accuracy: 0.39742 |  0:11:43s
epoch 108| loss: 0.72524 | train_balanced_accuracy: 0.642   |
valid_balanced_accuracy: 0.39709 |  0:11:49s
epoch 109| loss: 0.73933 | train_balanced_accuracy: 0.63137 |
valid_balanced_accuracy: 0.37966 |  0:11:55s
epoch 110| loss: 0.72338 | train_balanced_accuracy: 0.64403 |
valid_balanced_accuracy: 0.38482 |  0:12:02s
epoch 111| loss: 0.72852 | train_balanced_accuracy: 0.62995 |
valid_balanced_accuracy: 0.38325 |  0:12:08s
epoch 112| loss: 0.7201  | train_balanced_accuracy: 0.62729 |
valid_balanced_accuracy: 0.36838 |  0:12:15s
epoch 113| loss: 0.72654 | train_balanced_accuracy: 0.61102 |
valid_balanced_accuracy: 0.37809 |  0:12:21s
epoch 114| loss: 0.72222 | train_balanced_accuracy: 0.62514 |
valid_balanced_accuracy: 0.37412 |  0:12:28s
epoch 115| loss: 0.71867 | train_balanced_accuracy: 0.6302  |
valid_balanced_accuracy: 0.37747 |  0:12:34s
epoch 116| loss: 0.72453 | train_balanced_accuracy: 0.62727 |
valid_balanced_accuracy: 0.37794 |  0:12:40s
epoch 117| loss: 0.72526 | train_balanced_accuracy: 0.60387 |
valid_balanced_accuracy: 0.37015 |  0:12:47s
epoch 118| loss: 0.71324 | train_balanced_accuracy: 0.63403 |
valid_balanced_accuracy: 0.36995 |  0:12:53s
epoch 119| loss: 0.72227 | train_balanced_accuracy: 0.64801 |
valid_balanced_accuracy: 0.38905 |  0:13:00s
epoch 120| loss: 0.71987 | train_balanced_accuracy: 0.65199 |
valid_balanced_accuracy: 0.36964 |  0:13:07s
epoch 121| loss: 0.72485 | train_balanced_accuracy: 0.62439 |
valid_balanced_accuracy: 0.36822 |  0:13:13s
```

```
epoch 122| loss: 0.71824 | train_balanced_accuracy: 0.60993 |
valid_balanced_accuracy: 0.37306 |  0:13:20s
epoch 123| loss: 0.70235 | train_balanced_accuracy: 0.6301  |
valid_balanced_accuracy: 0.37618 |  0:13:26s
epoch 124| loss: 0.7161  | train_balanced_accuracy: 0.61837 |
valid_balanced_accuracy: 0.35595 |  0:13:33s
epoch 125| loss: 0.72457 | train_balanced_accuracy: 0.63801 |
valid_balanced_accuracy: 0.36894 |  0:13:39s
epoch 126| loss: 0.71504 | train_balanced_accuracy: 0.63463 |
valid_balanced_accuracy: 0.37613 |  0:13:46s

Early stopping occurred at epoch 126 with best_epoch = 6 and
best_valid_balanced_accuracy = 0.4123
```

[ ]:
```
# plot losses
# plt.plot(clf1_nopreproc.history['valid_logloss'])
```

[ ]:
```
# plot accuracy
plt.plot(clf1_nopreproc.history['train_balanced_accuracy'])
plt.plot(clf1_nopreproc.history['valid_balanced_accuracy'])
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f06ecddb790>]
```



[ ]:
```
# Saving the file
import pickle
```

```
temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
 ↪model_files/Tabnet'
# clf1_nopreproc.save(temp_file_path + '/Tabnet_model.h5')  # creates a HDF5␣
 ↪file 'my_model.h5'

with open(temp_file_path + '/tabnet_v2.pkl', 'wb') as f:  # Python 3: open(...,␣
 ↪'wb')
    pickle.dump(clf1_nopreproc, f)
```

[ ]:
```
nn_pred= clf1_nopreproc.predict(norm_test_df.fillna(0).values)
```

[ ]:
```
nn_pred.value_counts()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-257-d3a82199df5b> in <module>
----> 1 nn_pred.value_counts()

AttributeError: 'numpy.ndarray' object has no attribute 'value_counts'
```

[ ]:
```
df = pd.DataFrame(nn_pred)
nn_pred = df.idxmax(axis=1)
```

[ ]:
```
generate_model_report(test_target, nn_pred, 'micro')
generate_model_report(test_target, nn_pred, 'macro')
generate_model_report(test_target, nn_pred, 'weighted')
```

[ ]:
```
print('\nClassification Report\n')
print(classification_report(test_target, nn_pred))
```

[ ]:
```
# Creating  a confusion matrix,which compares the y_test and y_pred

cm = confusion_matrix(test_target, nn_pred)


# Creating a dataframe for a array-formatted Confusion matrix,so it will be␣
 ↪easy for plotting.
cm_df = pd.DataFrame(cm,
                     index = ['below_0','above_0','above_market'],
                     columns = ['below_0','above_0','above_market'])


#Plotting the confusion matrix
plt.figure(figsize=(10,10))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
```

```
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')

plt.show()
```

End of the Notebook

[ ]:
```

```

5.8 Random forest classifier

[ ]:
```
# fitting the model
model = RandomForestClassifier(n_estimators=500,␣
 ↪random_state=1,max_depth=8,n_jobs=-1)
model.fit(norm_train_df.fillna(0), target)
```

[ ]:
```
RandomForestClassifier(max_depth=8, n_estimators=500, n_jobs=-1, random_state=1)
```

[ ]:
```
target_test_pred = model.predict(norm_test_df.fillna(0))
```

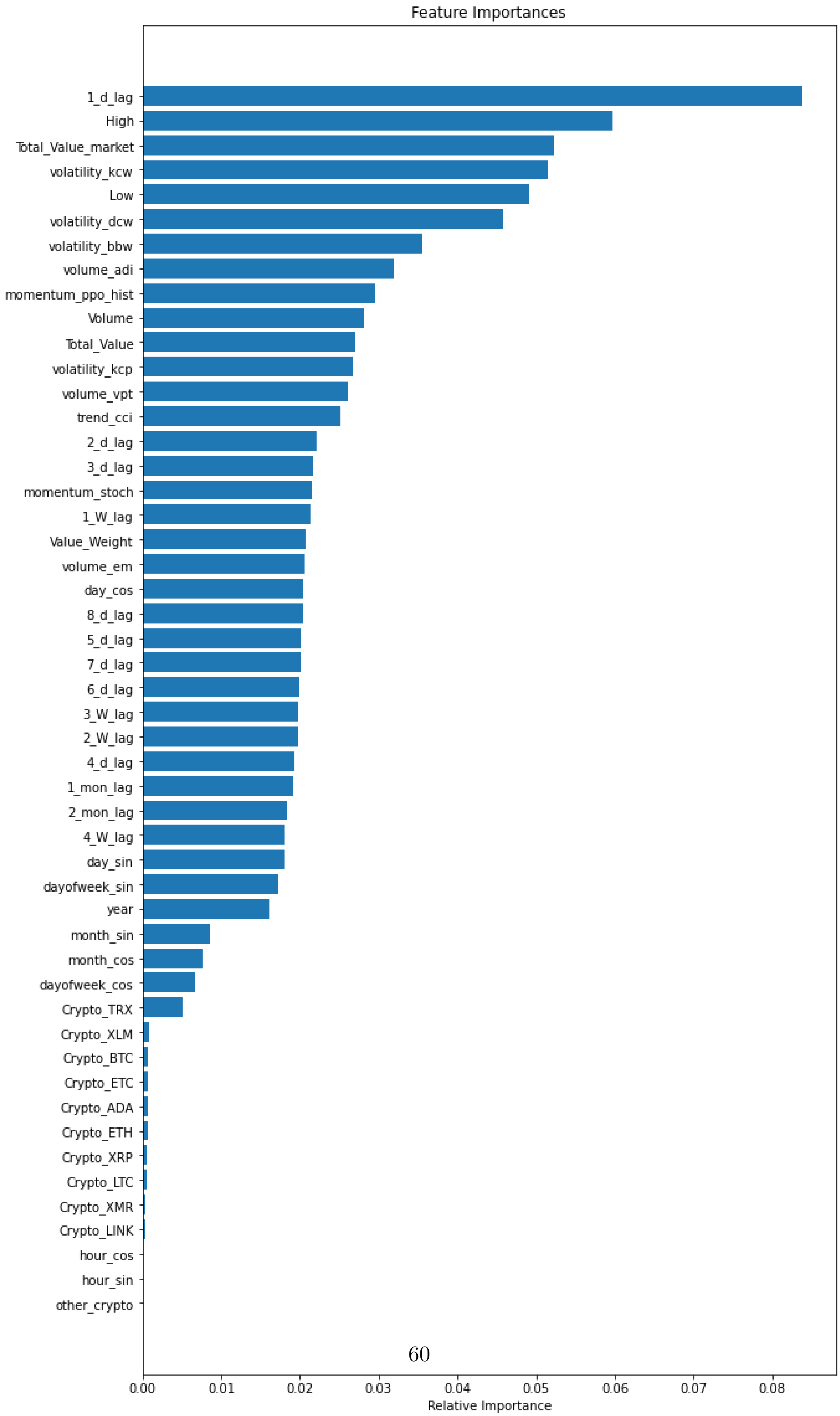
[ ]:
```
generate_model_report(test_target, target_test_pred, 'micro')
generate_model_report(test_target, target_test_pred, 'macro')
generate_model_report(test_target, target_test_pred, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.506
Precision =  0.506
Recall =  0.506
F1 Score =  0.506
=====================================
==================Printing the macro metrics===================
Precision =  0.338
Recall =  0.354
F1 Score =  0.345
=====================================
==================Printing the weighted metrics===================
Precision =  0.485
Recall =  0.506
F1 Score =  0.494
=====================================
```

[ ]:
```
# Feature importance
features=norm_train_df.columns
importances = model.feature_importances_
indices = np.argsort(importances)

plt.figure(figsize=(10,20))
plt.title('Feature Importances')
```

```python
plt.barh(range(len(indices)), importances[indices])
plt.yticks(range(len(indices)), features[indices])
plt.xlabel('Relative Importance')
```

```
[ ]: Text(0.5, 0, 'Relative Importance')
```



Feature Importances
1_d_lag
High
Total_Value_market
volatility_kcw
Low
volatility_dcw
volatility_bbw
volume_adi
momentum_ppo_hist
Volume
Total_Value
volatility_kcp
volume_vpt
trend_cci
2_d_lag
3_d_lag
momentum_stoch
1_W_lag
Value_Weight
volume_em
day_cos
8_d_lag
5_d_lag
7_d_lag
6_d_lag
3_W_lag
2_W_lag
4_d_lag
1_mon_lag
2_mon_lag
4_W_lag
day_sin
dayofweek_sin
year
month_sin
month_cos
dayofweek_cos
Crypto_TRX
Crypto_XLM
Crypto_BTC
Crypto_ETC
Crypto_ADA
Crypto_ETH
Crypto_XRP
Crypto_LTC
Crypto_XMR
Crypto_LINK
hour_cos
hour_sin
other_crypto
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Relative Importance

[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

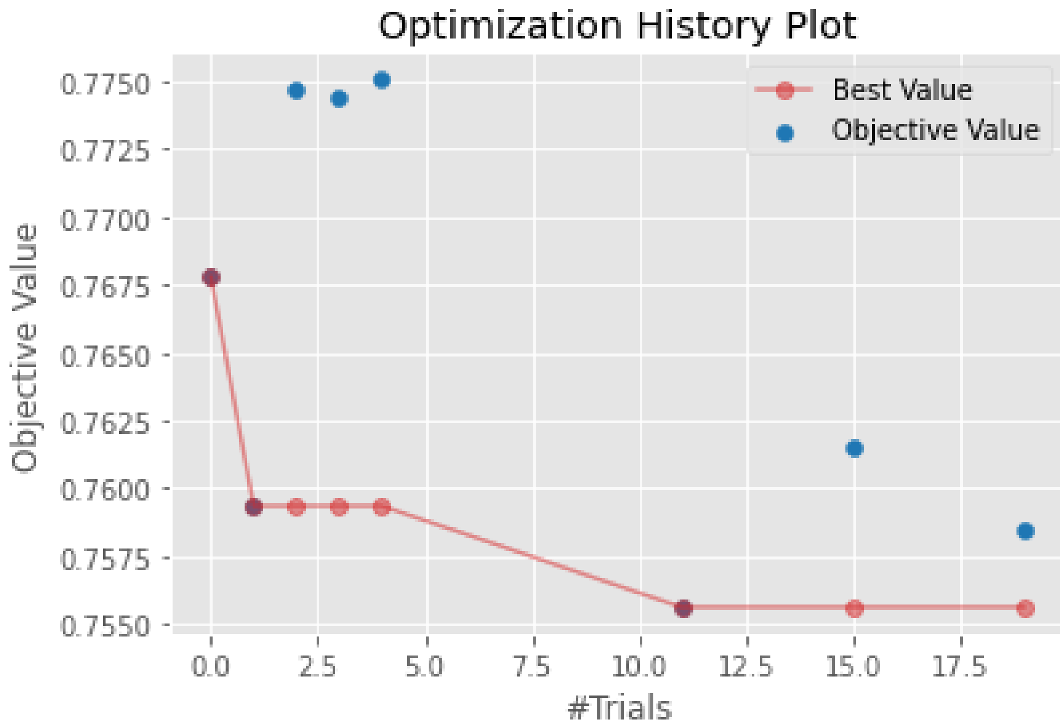
5.9 LGBM tuning with optuna

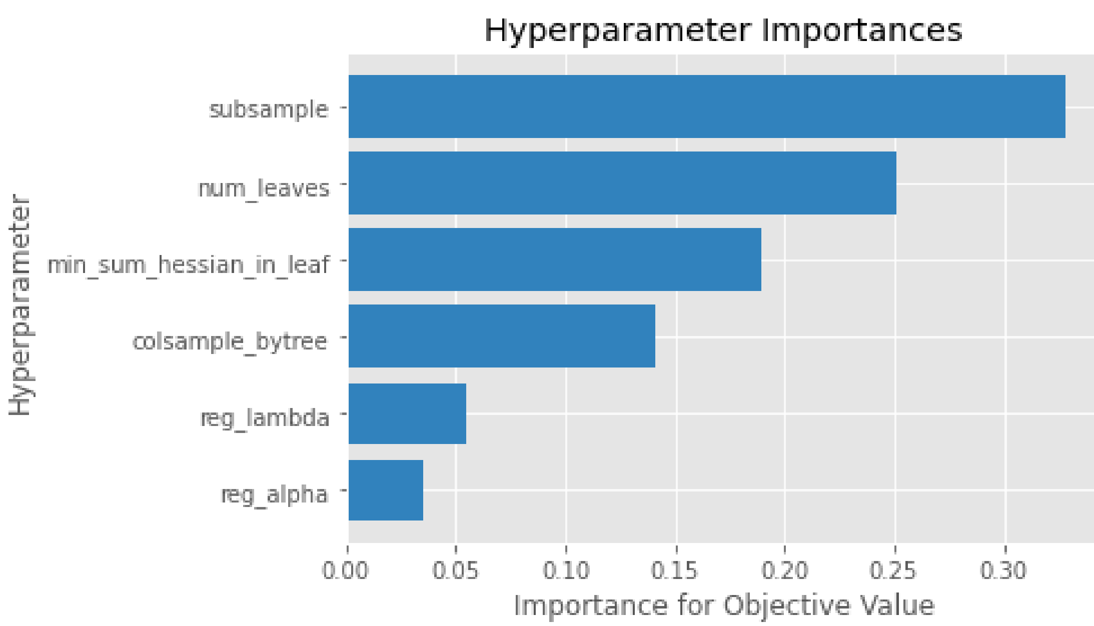
[ ]:

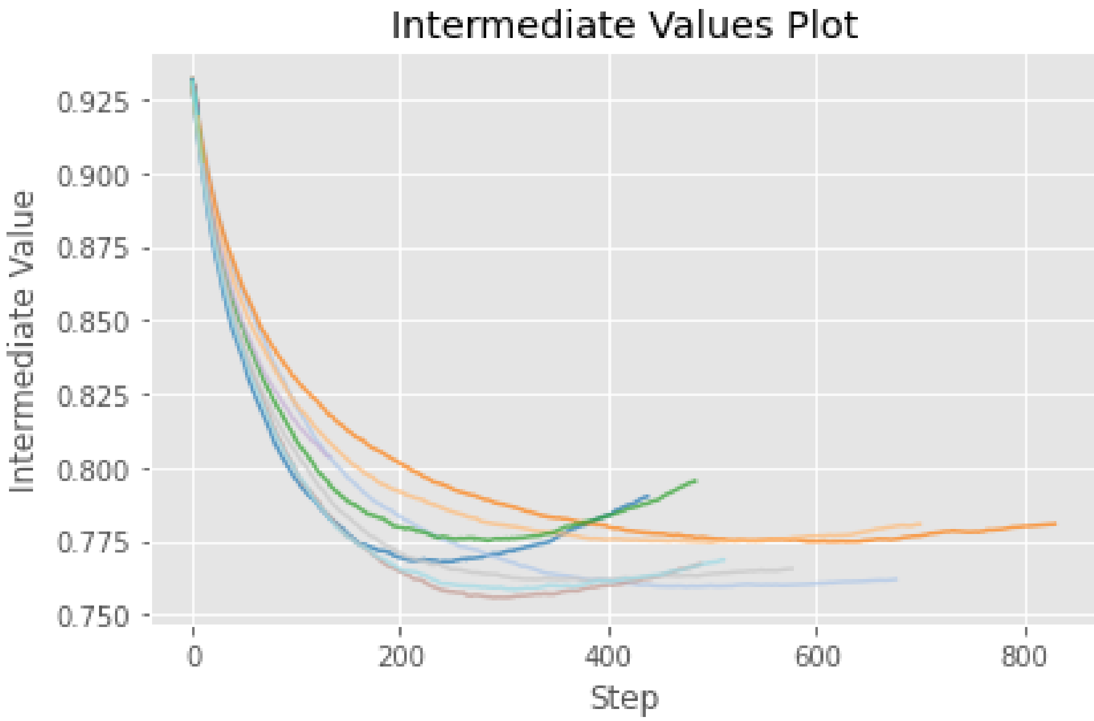
```python
from verstack import LGBMTuner
tuned_lgbm = LGBMTuner(metric = 'f1_macro', trials = 20)
tuned_lgbm.fit(norm_train_df.fillna(0), target)
```

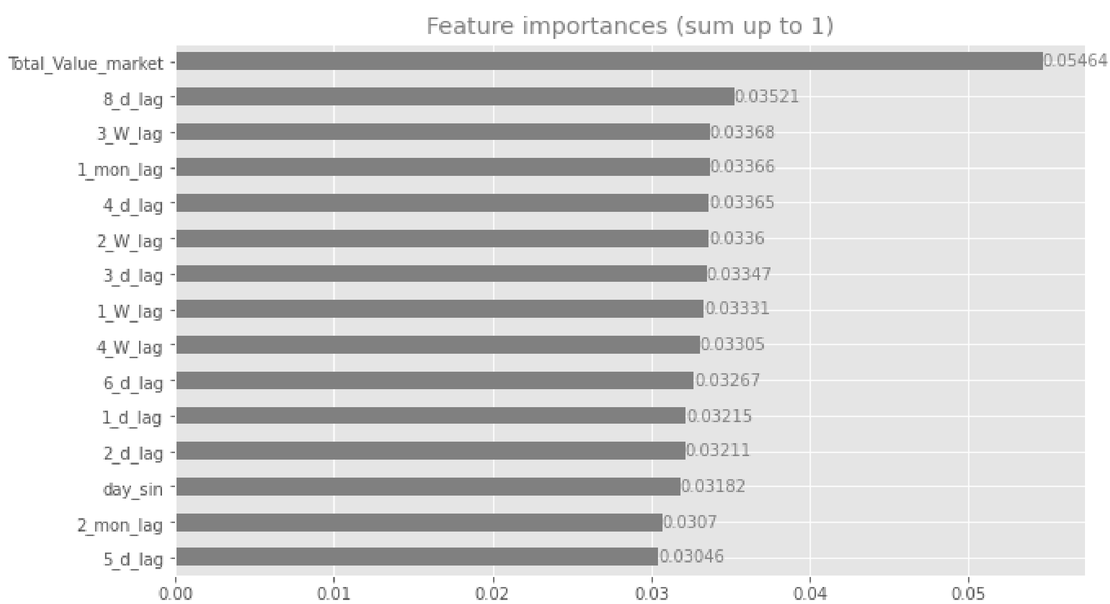
```
* Initiating LGBMTuner.fit
    . Settings:
    .. Trying 20 trials
    .. Evaluation metric: f1_macro
    .. Study direction: minimize log_loss

    . Trial number: 0 finished
    .. Optimization score (lower-better): log_loss: 0.7677924794265247
    .. Evaluation score (greater-better): f1_macro: 0.5203214698753702
…
    . Trial number: 1 finished
    .. Optimization score (lower-better): log_loss: 0.7593523232768736
    .. Evaluation score (greater-better): f1_macro: 0.5097983602870538
…
    . Trial number: 2 finished
    .. Optimization score (lower-better): log_loss: 0.774663563326048
    .. Evaluation score (greater-better): f1_macro: 0.5199973165458244
…
    . Trial number: 3 finished
    .. Optimization score (lower-better): log_loss: 0.774393559693144
    .. Evaluation score (greater-better): f1_macro: 0.5144030968450225
…
    . Trial number: 4 finished
    .. Optimization score (lower-better): log_loss: 0.7750818256537647
    .. Evaluation score (greater-better): f1_macro: 0.5209124315021884
…
    . Trial number: 11 finished
    .. Optimization score (lower-better): log_loss: 0.7556247573923124
    .. Evaluation score (greater-better): f1_macro: 0.5387465284643207
…
    . Trial number: 15 finished
    .. Optimization score (lower-better): log_loss: 0.761536687537761
```

```
    .. Evaluation score (greater-better): f1_macro: 0.5309296563380371
 …
    . Trial number: 19 finished
    .. Optimization score (lower-better): log_loss: 0.7584679418238669
    .. Evaluation score (greater-better): f1_macro: 0.5328100292785513
 …

   - Tune n_estimators with early_stopping
Training until validation scores don't improve for 200 rounds
[100]   train's multi_logloss: 0.501566 valid's multi_logloss: 0.798502
[200]   train's multi_logloss: 0.313777 valid's multi_logloss: 0.765368
[300]   train's multi_logloss: 0.208219 valid's multi_logloss: 0.758456
[400]   train's multi_logloss: 0.142102 valid's multi_logloss: 0.759817
[500]   train's multi_logloss: 0.0987624        valid's multi_logloss: 0.767103
Early stopping, best iteration is:
[319]   train's multi_logloss: 0.193157 valid's multi_logloss: 0.757329

   - Fitting optimized model with the follwing params:
task                              : train
learning_rate                     : 0.02
num_leaves                        : 177
colsample_bytree                  : 0.6277217293825994
subsample                         : 0.9154787443696969
bagging_freq                      : 1
max_depth                         : -1
verbosity                         : -1
reg_alpha                         : 1.2005946274475368e-08
reg_lambda                        : 1.210727364048483e-08
min_split_gain                    : 0.0
zero_as_missing                   : False
max_bin                           : 255
min_data_in_bin                   : 3
random_state                      : 42
num_classes                       : 3
objective                         : multiclass
metric                            : multi_logloss
num_threads                       : 0
min_sum_hessian_in_leaf           : 0.1610128003061735
n_estimators                      : 319
```


Optimization History Plot
Best Value
Objective Value
Objective Value
0.7750
0.7725
0.7700
0.7675
0.7650
0.7625
0.7600
0.7575
0.7550
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
#Trials


Hyperparameter Importances
Hyperparameter
subsample
num_leaves
min_sum_hessian_in_leaf
colsample_bytree
reg_lambda
reg_alpha
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Importance for Objective Value






```
. Optuna hyperparameters optimization finished
.. Best trial number:11    |     log_loss:         0.7556247573923124
----------------------------------------------------------------------------
```

```
 . n_estimators optimization finished
 .. best iteration:  319   |     multi_logloss:          0.7573285095803589
===========================================================================

Time elapsed for fit execution: 6 min 3.876 sec
```

[ ]:
```
tuned_lgbm.best_params
```

[ ]:
```
{'task': 'train',
 'learning_rate': 0.02,
 'num_leaves': 177,
 'colsample_bytree': 0.6277217293825994,
 'subsample': 0.9154787443696969,
 'bagging_freq': 1,
 'max_depth': -1,
 'verbosity': -1,
 'reg_alpha': 1.2005946274475368e-08,
 'reg_lambda': 1.210727364048483e-08,
 'min_split_gain': 0.0,
 'zero_as_missing': False,
 'max_bin': 255,
 'min_data_in_bin': 3,
 'random_state': 42,
 'num_classes': 3,
 'objective': 'multiclass',
 'metric': 'multi_logloss',
 'num_threads': 0,
 'min_sum_hessian_in_leaf': 0.1610128003061735,
 'n_estimators': 319}
```

[ ]:
```
tuned_lgbm
```

[ ]:
```
LGBMTuner(Evaluation metric: f1_macro
          trials: 20
          refit: True
          verbosity: 1
          visualization: True)
```

[ ]:
```
tuned_lgbm_pred = tuned_lgbm.predict(norm_test_df)
```

[ ]:
```
generate_model_report(test_target, tuned_lgbm_pred, 'micro')
generate_model_report(test_target, tuned_lgbm_pred, 'macro')
generate_model_report(test_target, tuned_lgbm_pred, 'weighted')
```

```
==================Printing the micro metrics===================
Accuracy =  0.518
Precision =  0.518
Recall =  0.518
```

```
F1 Score =  0.518
=====================================
==================Printing the macro metrics====================
Precision =  0.433
Recall =  0.379
F1 Score =  0.376
=====================================
==================Printing the weighted metrics====================
Precision =  0.517
Recall =  0.518
F1 Score =  0.501
=====================================
```

[ ]:
```
# Saving the file
import pickle
temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/LGBM_hourly'
with open(temp_file_path + '/tuned_lgbm_f13_final.pkl', 'wb') as f:  # Python 3:
↪ open(..., 'wb')
    pickle.dump(tuned_lgbm, f)
```

5.10 LGBM After Optimisation

[ ]:
```
import lightgbm as lgb

untuned_lgbm = lgb.LGBMClassifier( learning_rate= 0.02,
 num_leaves= 185,
 colsample_bytree= 0.9973006775338719,
 subsample= 0.8626362747879762,
 bagging_freq= 1,
 max_depth= -1,
 verbosity= -1,
 reg_alpha= 0.6849391813640976,
 reg_lambda= 1.4776089934723414e-08,
 min_split_gain= 0.0,
 zero_as_missing= False,
 max_bin= 255,
 min_data_in_bin= 3,
 random_state= 42,
 num_classes= 3,
 objective= 'multiclass',
 metric= 'multi_logloss',
 num_threads= 0,
 min_sum_hessian_in_leaf= 8.698870626111832,
 n_estimators= 780)
untuned_lgbm.fit(norm_train_df.fillna(0), target)
```

```
[LightGBM] [Warning] num_threads is set=0, n_jobs=-1 will be ignored. Current
```

```
value: num_threads=0
[LightGBM] [Warning] min_sum_hessian_in_leaf is set=8.698870626111832,
min_child_weight=0.001 will be ignored. Current value:
min_sum_hessian_in_leaf=8.698870626111832
[LightGBM] [Warning] bagging_freq is set=1, subsample_freq=0 will be ignored.
Current value: bagging_freq=1
```

[ ]:
```
LGBMClassifier(bagging_freq=1, colsample_bytree=0.9973006775338719,
               learning_rate=0.02, max_bin=255, metric='multi_logloss',
               min_data_in_bin=3, min_sum_hessian_in_leaf=8.698870626111832,
               n_estimators=780, num_classes=3, num_leaves=185, num_threads=0,
               objective='multiclass', random_state=42,
               reg_alpha=0.6849391813640976, reg_lambda=1.4776089934723414e-08,
               subsample=0.8626362747879762, verbosity=-1,
               zero_as_missing=False)
```

[ ]:
```
lgbm_pred=untuned_lgbm.predict(norm_test_df)
```

[ ]:

5.10.1 Test Metrics

[ ]:
```
generate_model_report(test_target, lgbm_pred, 'micro')
generate_model_report(test_target, lgbm_pred, 'macro')
generate_model_report(test_target, lgbm_pred, 'weighted')
```

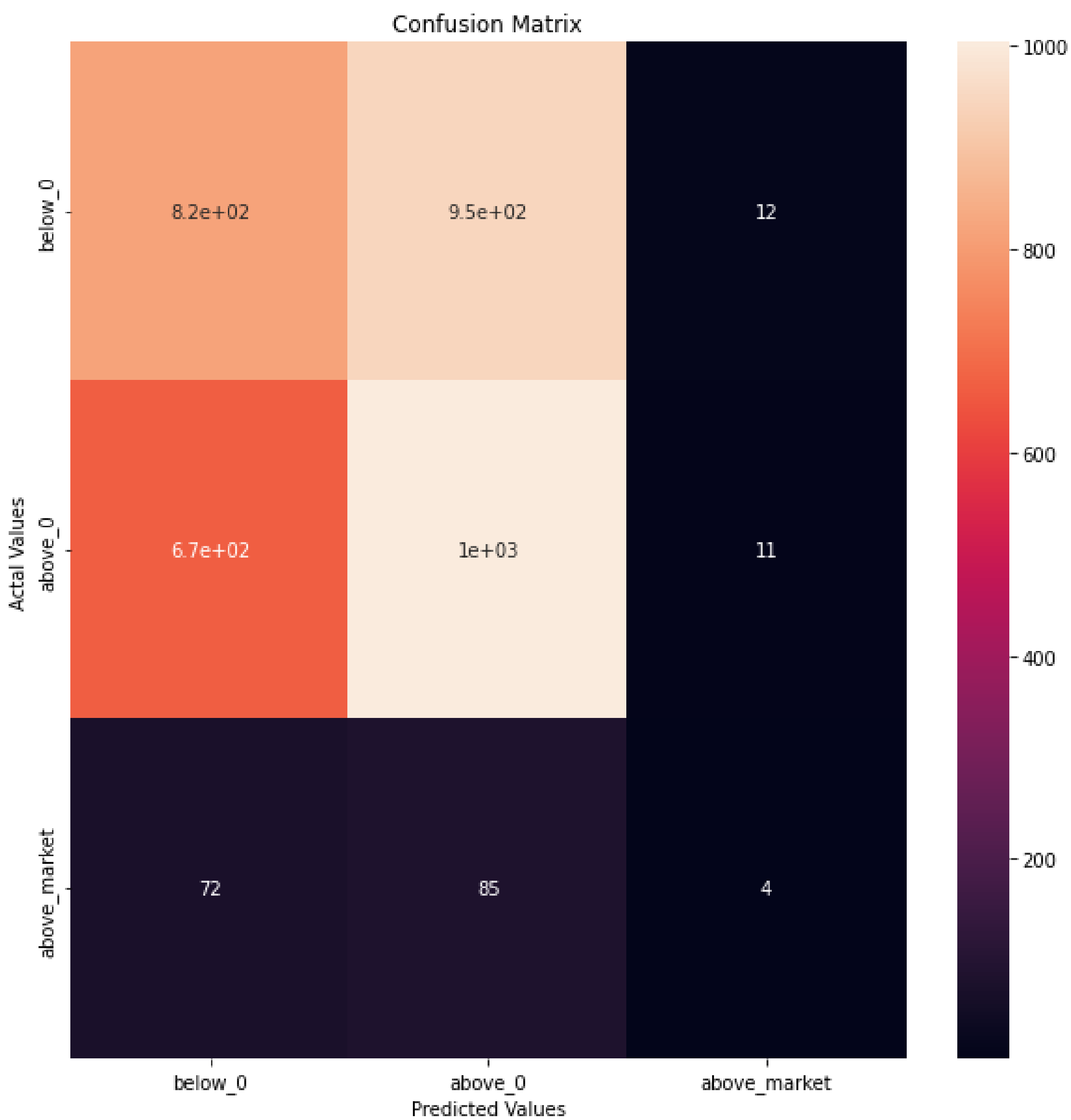
```
==================Printing the micro metrics===================
Accuracy =  0.508
Precision =  0.508
Recall =  0.508
F1 Score =  0.508
=====================================
==================Printing the macro metrics===================
Precision =  0.402
Recall =  0.391
F1 Score =  0.385
=====================================
==================Printing the weighted metrics===================
Precision =  0.512
Recall =  0.508
F1 Score =  0.495
=====================================
```

5.10.2 Confusion matrix

[ ]:
```
from sklearn.metrics import classification_report
print('\nClassification Report\n')
print(classification_report(test_target, lgbm_pred))
```

```
Classification Report

              precision    recall  f1-score   support

           0       0.55      0.38      0.45      1778
           1       0.50      0.68      0.58      1682
           2       0.15      0.11      0.13       161

    accuracy                           0.51      3621
   macro avg       0.40      0.39      0.39      3621
weighted avg       0.51      0.51      0.49      3621
```

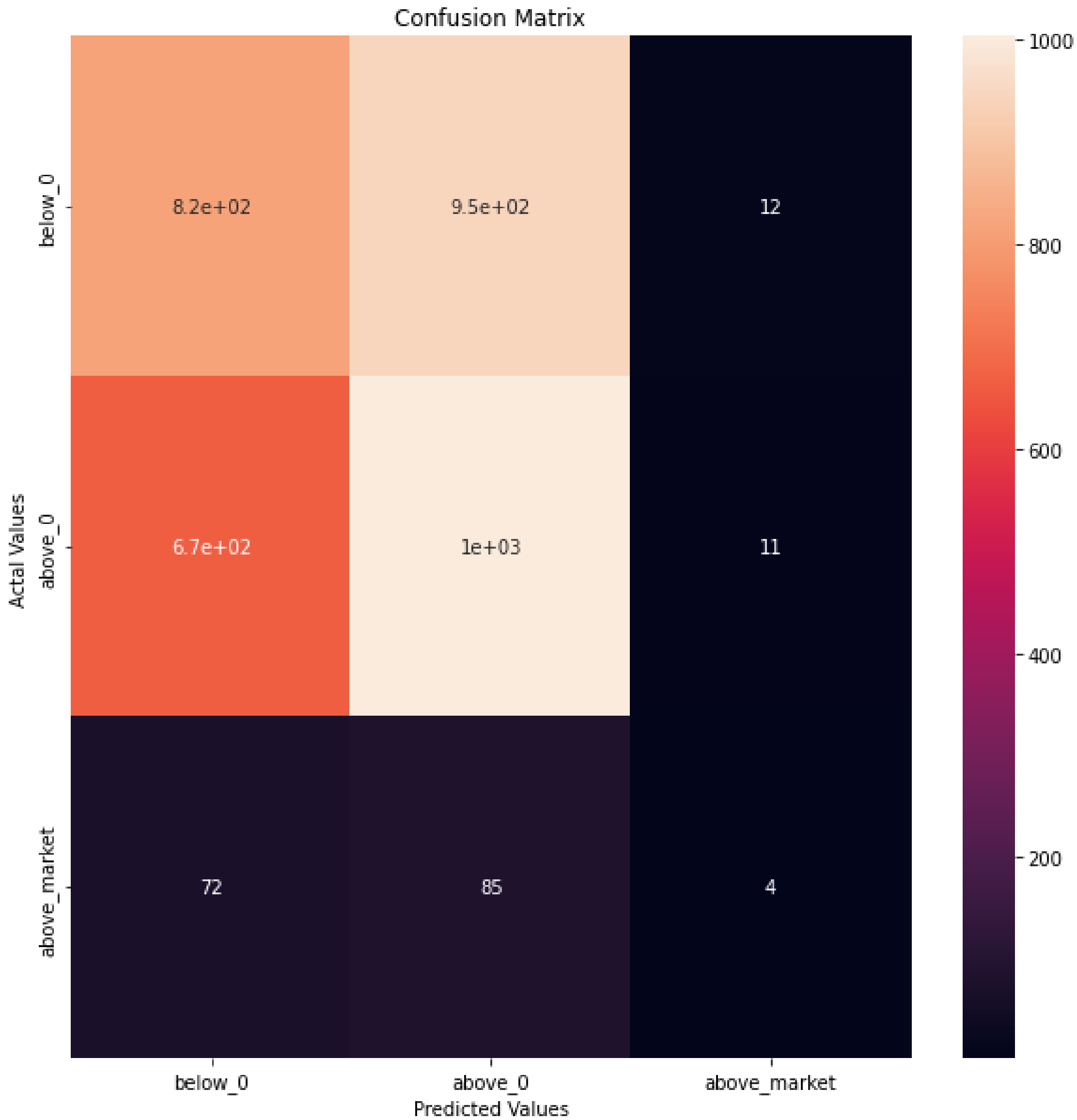
[ ]:
```
# Creating  a confusion matrix,which compares the y_test and y_pred
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test_target, lgbm_pred)


# Creating a dataframe for a array-formatted Confusion matrix,so it will be␣
 ↪easy for plotting.
cm_df = pd.DataFrame(cm,
                     index = ['below_0','above_0','above_market'],
                     columns = ['below_0','above_0','above_market'])


#Plotting the confusion matrix
plt.figure(figsize=(10,10))
sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')

plt.show()
```




```
[ ]: # Creating a dataframe for a array-formatted Confusion matrix,so it will be␣
     ↪easy for plotting.
     cm_df = pd.DataFrame(cm,
                          index = ['below_0','above_0','above_market'],
                          columns = ['below_0','above_0','above_market'])
```

```
[ ]: #Plotting the confusion matrix
     plt.figure(figsize=(10,10))
     sns.heatmap(cm_df, annot=True)
     plt.title('Confusion Matrix')
     plt.ylabel('Actal Values')
```

```
plt.xlabel('Predicted Values')

plt.show()
```




```
[ ]: # Saving the file
import pickle
temp_file_path = '/content/drive/MyDrive/MADS_23_DL_final_project/data/
↪model_files/LGBM_hourly'
with open(temp_file_path + '/Tuned_lgbm_v13_final2.pkl', 'wb') as f:  # Python␣
↪3: open(..., 'wb')
    pickle.dump(untuned_lgbm, f)
```

6 End of the Notebook