Predicting Solar Energetic Particles Using SDO/HMI Vector Magnetic Data Products and a Bidirectional LSTM Network

Table of Contents

Author(s)

- Author1 = {"name": "Yasser Abduallah", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "ya54@njit.edu", "orcid": "https://orcid.org/0000-0003-0792-2270%22%7D
- Author2 = {"name": "Vania K. Jordanova", "affiliation": "Space Science and Applications, Los Alamos National Laboratory, Los Alamos, NM", "email": "vania@lanl.gov", "orcid": "https://orcid.org/0000-0003-0475-8743%22%7D

- Author3 = {"name": "Hao Liu", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "hl422@njit.edu", "orcid": "https://orcid.org/0000-0002-1975-1272%22%7D
- Author4 = {"name": " Qin Li", "affiliation": "Big Bear Solar Observatory, New Jersey Institute of Technology, 40386 North Shore Lane, Big Bear City, CA 92314, USA", "email": "ql47@njit.edu", "orcid": "https://orcid.org/0000-0002-6178-7471%22%7D
- Author5 = {"name": "Jason T. L. Wang", "affiliation": "Department of Computer Science, New Jersey Institute of Technology", "email": "wangj@njit.edu", "orcid": "https://orcid.org/0000-0002-2486-1097%22%7D
- Author6 = {"name": "Haimin Wang", "affiliation": "Institute for Space Weather Sciences, New Jersey Institute of Technology", "email": "haimin.wang@njit.edu", "orcid": "https://orcid.org/0000-0002-5233-565X%22%7D

Purpose

Solar energetic particles (SEPs) are an essential source of space radiation, and are hazardous for humans in space, spacecraft, and technology in general. We provide a deep-learning method, specifically a bidirectional long short-term memory (biLSTM) network, to predict if an active region (AR) would produce an SEP event given that (i) the AR will produce an M- or X-class flare and a coronal mass ejection (CME) associated with the flare, or (ii) the AR will produce an M- or X-class flare regardless of whether or not the flare is associated with a CME.

We aim to solve the following two binary prediction problems. [FC_S problem] Given a data sample x_t at time point t in an AR where the AR will produce an M- or X-class flare within the next *T* hours of t and the flare initiates a CME, we predict whether x_t is positive or negative. Predicting x_t to be positive means that the AR will produce an SEP event associated with the flare/CME. Predicting x_t to be negative means that the AR will not produce an SEP event associated with the flare/CME. [F_S problem] Given a data sample x_t at time point t in an AR where the AR will produce an M- or X-class flare within the next *T* hours of t regardless of whether or not the flare initiates a CME, we predict whether x_t is positive or negative. Predicting x_t to be positive means that the AR will produce an SEP event associated with the flare. Predicting x_t to be negative means that the AR will not produce an SEP event associated with the flare. For both of the two binary prediction problems, we consider *T* ranging from 12 to 72 in 12 hr intervals.

In this notebook we provide an overview of the BiLSTM system to demonstrate how to predict SEP using deep learning (DL) and SDO/HMI vector magnetic data products (SHARP parameters).

Technical Contributions

- We provide the community with a new tool to predict the occurrance of solar energetic particles (SEP) for the next 12, 24, 36, 48, 60, or 72 hours ahead.

Methodology

Here we present a prediction system, named BiLSTM, for predicting solar energetic particles using deep learning (DL) based on HMI's vector magnetic data products. The data samples used in this study are collected from the Geostationary Operational Environmental Satellite's X-ray flare catalogs provided by the National Centers for Environmental Information. We select M- and X-class flares with identified ARs in the catalogs for the period between 2010 and 2021, and find the associations of flares, CMEs, and SEPs in the Space Weather Database of Notifications, Knowledge, Information during the same period. Each data sample contains physical parameters collected from the Helioseismic and Magnetic Imager on board the Solar Dynamics Observatory. Specifically, we collected SHARP data samples from the data series, hmi.sharp_cea_720s, using the Python package SunPy at a cadence of 12 minutes. In collecting the data samples, we focused on 18 physical parameters previously used for SEP predictions. Appropriately labeling the data samples is crucial for machine learning. We surveyed M- and X-class flares that occurred between 2010 and 2021 with identified ARs in the GOES X-ray flare catalogs provided by the National Centers for Environmental Information (NCEI).

To solve the binary prediction problems. Consider the FC_S problem where *T* = 24 hr. Here we want to predict whether a given test data sample x_t at time point t is positive (blue) or negative (green) given that there will be an M- or X-class flare within the next 24 hr of t, and the flare initiates a CME. If there is an SEP event associated with the flare/CME, and we predict x_t to be positive (blue), then this is a correct prediction as illustrated in Figure 1(c). If there is an SEP event associated with the flare/CME, but we predict x_t to be negative (green), then this is a wrong prediction as illustrated in Figure 1(e). On the other hand, if there is no SEP event associated with the flare/CME, and we predict x_t to be negative (green), then this is a correct prediction as illustrated in Figure 1(d). If there is no SEP event associated with the flare/CME, but we predict x_t to be positive (blue), then this is a wrong prediction as illustrated in Figure 1(f). The F_S problem is solved similarly.

The architecture of the BiLSTM network is shown in Figure 2 and described as follow. Yellow boxes represent biLSTM cells. These cells are connected to an attention layer (A) that contains m neurons, which are connected to a fully connected layer (FCL). (In the study presented here, m is set to 10.) During testing/prediction, the input to the network is a test data sequence with m consecutive data samples x_{t-m+1}, x_{t-m+2} ... x_{t-1}, x_t where x_t is the test data sample at time point t. The trained BiLSTM network predicts the label (color) of the test data sequence, more precisely the label (color) of x_t . The output layer of the BiLSTM network calculates a probability ($\hat{y}$) between 0 and 1. If $\hat{y}$ is greater than or equal to a threshold, which is set to 0.5, the BiLSTM network outputs 1 and predicts x_t to be positive, i.e., predicts the label (color) of x_t to be blue; see Figure 1. Otherwise, the BiLSTM network outputs 0 and predicts x_t to be negative, i.e., predicts the label (color) of x_t to be green; see Figure 1.

This notebook leverages python deep learning to describe the steps on how to use the BiLSTM tool to predict the binary SEP classification problems we are sovling.

Funding

This work was supported by U.S. NSF grants AGS-1927578 and AGS-1954737 and supported by NASA under grants 80NSSC18K1705, 80NSSC19K0068, and 80NSSC20K1282.

Keywords

keywords=["Flare", "Prediction", "Machine", "Learning", "SHARP","Solar",'Particles","Energetic","Coronal","Mass","Ejection"]

Citation

To cite this notebook: Yasser Abduallah, Vania K. Jordanova, Hao Liu, Qin Li, Jason T. L. Wang, and Haimin Wang. Predicting Solar Energetic Particles Using SDO/HMI Vector Magnetic Data Products and a Bidirectional LSTM Network, available at: https://github.com/ccsc-tools/SEP-prediction/blob/main/YA_01_PredictingSEPUsingBiLSTM.ipynb.

Acknowledgements

We thank the team of SDO/HMI for producing vector magnetic data products. The flare catalogs were prepared by and made available through NOAA NCEI. And we also thank Manolis K. Georgoulis for helpful conversations in the SHINE 2019 Conference.

Setup

Installation on Local Machine

Running this notebook in a local machine requires Python version 3.9.x with the following packages and their version:

Library	Version	Description
keras	2.6.0	Deep learning API
numpy	1.19.5	Array manipulation
scikit-learn	1.0.1	Machine learning
pandas	1.4.1	Data loading and manipulation
tensorboard	2.8.0	Provides the visualization and tooling needed for machine learning
tensorflow-gpu	2.6.0	Deep learning tool for high performance computation
tensorflow-estimator	2.8.0	A high-level TensorFlow API. Estimators encapsulate the training, prediction, and evalution functions
matplotlib	3.4.3	Visualization and charts required for notebook only

You may install the package using Python pip packages manager as follows:

pip install tensorflow-gpu==2.6.0 tensorflow-estimator==2.8.0 numpy==1.19.5 pandas==1.4.1 keras==2.6.0 scikit-learn==1.0.1 matplotlib==3.4.3

Library Import

The following libraries need to be imported.

In [1]:
```
#supress warning messages
import warnings
warnings.filterwarnings('ignore')
print('Importing packages..')
# Data manipulation
import pandas as pd
import numpy as np
import os
print('Packages imported')
#make sure the scripts are executed in the correct pacakage location.
if os.path.exists('SEP_Package'):
    print('Changing working directory to SEP_Package..')
    os.chdir('SEP_Package')
import sys
sys.path.append('.')
```

```
Importing packages..
Packages imported
Changing working directory to SEP_Package..
```

Data Processing and Analysis

We collected and extracted information from NASA's Space Weather Database of Notifications, Knowledge, Information (DONKI) to tag, for any given M- or X-class flare, whether it produced a CME and/or SEP event. We cross-checked the flare records in DONKI and GOES X-ray flare catalogs to ensure that each flare record was associated with an AR; otherwise the flare record was removed from our study.

We then created two databases of ARs for the period between 2010 and 2021. ARs from 2010, 2016, and 2018–2021 were excluded from the study due to the lack of qualified data samples or the absence of SEP events associated with M-/X-class flares and CMEs. Thus, the databases contain ARs from six years, namely 2011–2015 and 2017. In our first database, referred to as the FC_S database, each record corresponds to an AR, contains an M- or X-class flare as well as a CME associated with the flare, and is tagged by whether the flare/CME produce an SEP event. In this database, there are 31 records tagged by "yes" indicating they are associated with SEP events while there are 97 records tagged by "no" indicating they are not associated with SEP events. In our second database, referred to as the F_S database, each record corresponds to an AR, contains an M- or X-class flare, and is tagged by whether the flare produces an SEP event regardless of whether or not the flare initiates a CME. In this database, there are 40 records tagged by "yes" indicating they are associated with SEP events while there are 700 records tagged by "no" indicating they are not associated with SEP events.

Binder

This notebook is Binder enabled and can be run on mybinder.org by using the image link below:

launch binder

Please note that starting Binder might take some time to create and start the image.

SEP BiLSTM Workflow and Results

Data Preparation and Loading

The data directory inside the SEP_Package folder includes all training and test data sets required to run the run the notebook.

- The CSV files start with events_fc_testing are used to test the first SEP prediction FC_S.
- The CSV files start with events_fc_training are used to train the first SEP prediction FC_S.
- The CSV files start with events_f_testing are used to train the second SEP prediction F_S.
- The CSV files start with events_f_training are used to train the second SEP prediction F_S.

The files are loaded and used during the testing and training process.

Predicting with Pretrained Models

There are default and pretrained models that can be used to predict without running your own trained model. The models_directory is set to default_models which uses all pretrained algorithms.

In [2]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in d:\jupyternotebooks\master_project\solar
\lib\site-packages (2.10.0)
Requirement already satisfied: typing-extensions>=3.6.6 in d:\jupyternotebooks\master
_project\solar\lib\site-packages (from tensorflow) (3.7.4.3)
Requirement already satisfied: libclang>=13.0.0 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from tensorflow) (14.0.6)
Requirement already satisfied: numpy>=1.20 in d:\jupyternotebooks\master_project\sola
r\lib\site-packages (from tensorflow) (1.23.4)
Requirement already satisfied: flatbuffers>=2.0 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from tensorflow) (22.9.24)
Requirement already satisfied: google-pasta>=0.1.1 in d:\jupyternotebooks\master_proj
ect\solar\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: setuptools in d:\jupyternotebooks\master_project\solar
\lib\site-packages (from tensorflow) (65.4.1)
Requirement already satisfied: keras<2.11,>=2.10.0 in d:\jupyternotebooks\master_proj
ect\solar\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: tensorboard<2.11,>=2.10 in d:\jupyternotebooks\master_
project\solar\lib\site-packages (from tensorflow) (2.10.1)
Requirement already satisfied: wrapt>=1.11.0 in d:\jupyternotebooks\master_project\so
lar\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: absl-py>=1.0.0 in d:\jupyternotebooks\master_project\s
olar\lib\site-packages (from tensorflow) (1.3.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in d:\jupyternote
books\master_project\solar\lib\site-packages (from tensorflow) (0.27.0)
Requirement already satisfied: astunparse>=1.6.0 in d:\jupyternotebooks\master_projec
t\solar\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in d:\jupyternotebooks\master_proj
ect\solar\lib\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in d:\jupyternotebooks\master_proj
ect\solar\lib\site-packages (from tensorflow) (1.49.1)
Requirement already satisfied: six>=1.12.0 in d:\jupyternotebooks\master_project\sola
r\lib\site-packages (from tensorflow) (1.15.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in d:\jupyternotebooks\mast
er_project\solar\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: packaging in d:\jupyternotebooks\master_project\solar
\lib\site-packages (from tensorflow) (21.3)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in d:\jupyternotebooks\master_pr
oject\solar\lib\site-packages (from tensorflow) (3.19.6)
Requirement already satisfied: h5py>=2.9.0 in d:\jupyternotebooks\master_project\sola
r\lib\site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: opt-einsum>=2.3.2 in d:\jupyternotebooks\master_projec
t\solar\lib\site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: termcolor>=1.1.0 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: tensorflow-estimator<2.11,>=2.10.0 in d:\jupyternotebo
oks\master_project\solar\lib\site-packages (from tensorflow) (2.10.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in d:\jupyternotebooks\master_proje
ct\solar\lib\site-packages (from astunparse>=1.6.0->tensorflow) (0.37.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in d:\jupyternotebooks\master_pr
oject\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.12.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in d:\jupyternotebooks\m
aster_project\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (1.
8.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in d:\jupyternot
ebooks\master_project\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorfl
ow) (0.6.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in d:\jupyternotebook
```

```
s\master_project\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow)
(0.4.6)
Requirement already satisfied: requests<3,>=2.21.0 in d:\jupyternotebooks\master_proj
ect\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.28.1)
Requirement already satisfied: markdown>=2.6.8 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (3.4.1)
Requirement already satisfied: werkzeug>=1.0.1 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from tensorboard<2.11,>=2.10->tensorflow) (2.2.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in d:\jupyternotebooks\master
_project\solar\lib\site-packages (from packaging->tensorflow) (3.0.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in d:\jupyternotebooks\master_pr
oject\solar\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->t
ensorflow) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in d:\jupyternotebooks\master_project\so
lar\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->tensorflo
w) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in d:\jupyternotebooks\master_p
roject\solar\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.11,>=2.10->
tensorflow) (5.2.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in d:\jupyternotebooks\master
_project\solar\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<
2.11,>=2.10->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in d:\jupyternotebooks\master_
project\solar\lib\site-packages (from markdown>=2.6.8->tensorboard<2.11,>=2.10->tenso
rflow) (5.0.0)
Requirement already satisfied: idna<4,>=2.5 in d:\jupyternotebooks\master_project\sol
ar\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensorflow)
(3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in d:\jupyternotebooks\master_pr
oject\solar\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->ten
sorflow) (1.26.12)
Requirement already satisfied: charset-normalizer<3,>=2 in d:\jupyternotebooks\master
_project\solar\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->
tensorflow) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in d:\jupyternotebooks\master_proje
ct\solar\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.11,>=2.10->tensor
flow) (2022.9.24)
Requirement already satisfied: MarkupSafe>=2.1.1 in d:\jupyternotebooks\master_projec
t\solar\lib\site-packages (from werkzeug>=1.0.1->tensorboard<2.11,>=2.10->tensorflow)
(2.1.1)
Requirement already satisfied: zipp>=0.5 in d:\jupyternotebooks\master_project\solar
\lib\site-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.11,>
=2.10->tensorflow) (3.9.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in d:\jupyternotebooks\master_pro
ject\solar\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tens
orboard<2.11,>=2.10->tensorflow) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in d:\jupyternotebooks\master_project
\solar\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=
0.4.1->tensorboard<2.11,>=2.10->tensorflow) (3.2.1)
```

```
WARNING: You are using pip version 22.0.4; however, version 22.3 is available.
You should consider upgrading via the 'D:\jupyternotebooks\master_project\solar\Scrip
ts\python.exe -m pip install --upgrade pip' command.
```

In [3]:

```
#Test default models for FC_S for 12-72 hours.
from SEP_test import test

models_directory='default_models'
```

```
print('Test default models for first classification type FC_S and for all time windows
classification_type='FC_S'
starting_time_window=12
ending_time_window= 72
test(classification_type,starting_time_window,ending_time_window+1,models_directory=mc
```

```
SUCCESS: Found GPU: /device:GPU:0
Test default models for first classification type FC_S and for all time windows: 12 t
o 72 hours.
Running classification test type: FC_S  training for h = 12 hour ahead
testing data file: data/events_fc_testing_12.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_12hr
Building model for: default_models\sep_model_fc_12hr\model_weights
Loading weights from: default_models\sep_model_fc_12hr\model_weights
Loading data from data file: data/events_fc_testing_12.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_12.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_12.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 24 hour ahead
testing data file: data/events_fc_testing_24.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_24hr
Building model for: default_models\sep_model_fc_24hr\model_weights
Loading weights from: default_models\sep_model_fc_24hr\model_weights
Loading data from data file: data/events_fc_testing_24.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_24.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_24.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 36 hour ahead
testing data file: data/events_fc_testing_36.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_36hr
Building model for: default_models\sep_model_fc_36hr\model_weights
Loading weights from: default_models\sep_model_fc_36hr\model_weights
Loading data from data file: data/events_fc_testing_36.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_36.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_36.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 48 hour ahead
testing data file: data/events_fc_testing_48.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_48hr
Building model for: default_models\sep_model_fc_48hr\model_weights
Loading weights from: default_models\sep_model_fc_48hr\model_weights
Loading data from data file: data/events_fc_testing_48.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_48.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_48.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 60 hour ahead
testing data file: data/events_fc_testing_60.csv
```

```
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_60hr
Building model for: default_models\sep_model_fc_60hr\model_weights
Loading weights from: default_models\sep_model_fc_60hr\model_weights
Loading data from data file: data/events_fc_testing_60.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_60.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_60.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 72 hour ahead
testing data file: data/events_fc_testing_72.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_fc_72hr
Building model for: default_models\sep_model_fc_72hr\model_weights
Loading weights from: default_models\sep_model_fc_72hr\model_weights
Loading data from data file: data/events_fc_testing_72.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_72.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_FC_S_72.csv
-------------------------------------------------------
```

In [4]:

```
#Testing default models for F_S for 12-72 hours.
from SEP_test import test

models_directory='default_models'
print('Test default models for second classification type F_S and for all time windows
classification_type='F_S'
test(classification_type,starting_time_window,ending_time_window+1,models_directory=mc
```

```
Test default models for second classification type F_S and for all time windows: 12 t
o 72 hours.
Running classification test type: F_S  training for h = 12 hour ahead
testing data file: data/events_f_testing_12.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_f_12hr
Building model for: default_models\sep_model_f_12hr\model_weights
Loading weights from: default_models\sep_model_f_12hr\model_weights
Loading data from data file: data/events_f_testing_12.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_12.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_12.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 24 hour ahead
testing data file: data/events_f_testing_24.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_f_24hr
Building model for: default_models\sep_model_f_24hr\model_weights
Loading weights from: default_models\sep_model_f_24hr\model_weights
Loading data from data file: data/events_f_testing_24.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_24.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_24.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 36 hour ahead
testing data file: data/events_f_testing_36.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_f_36hr
Building model for: default_models\sep_model_f_36hr\model_weights
Loading weights from: default_models\sep_model_f_36hr\model_weights
Loading data from data file: data/events_f_testing_36.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_36.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_36.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 48 hour ahead
testing data file: data/events_f_testing_48.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_f_48hr
Building model for: default_models\sep_model_f_48hr\model_weights
Loading weights from: default_models\sep_model_f_48hr\model_weights
Loading data from data file: data/events_f_testing_48.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_48.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_48.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 60 hour ahead
testing data file: data/events_f_testing_60.csv
Loading the model and its weights.
```

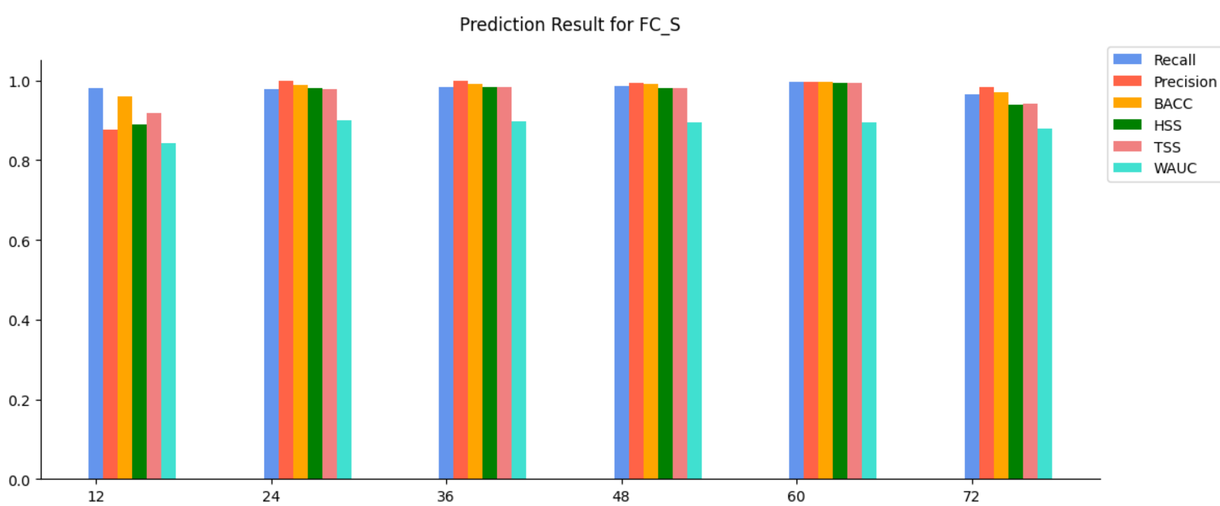
```
Loading weights from model dir: default_models\sep_model_f_60hr
Building model for: default_models\sep_model_f_60hr\model_weights
Loading weights from: default_models\sep_model_f_60hr\model_weights
Loading data from data file: data/events_f_testing_60.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_60.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_60.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 72 hour ahead
testing data file: data/events_f_testing_72.csv
Loading the model and its weights.
Loading weights from model dir: default_models\sep_model_f_72hr
Building model for: default_models\sep_model_f_72hr\model_weights
Loading weights from: default_models\sep_model_f_72hr\model_weights
Loading data from data file: data/events_f_testing_72.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_72.csv
Saving the performance metrics to files: default_results\SEP_performance_metrics_BiLS
TM_F_S_72.csv
-------------------------------------------------------
```

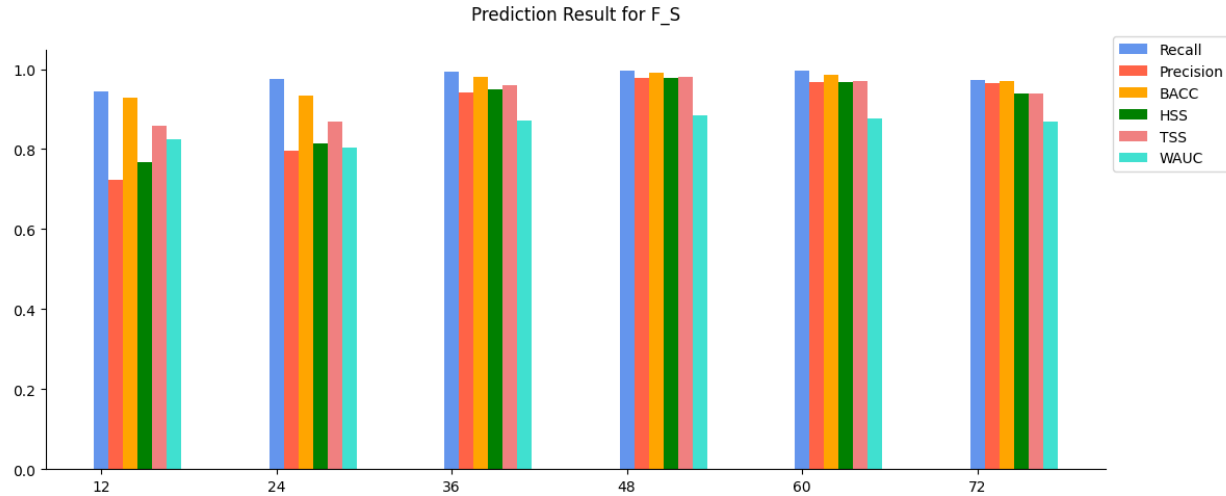
Plotting the Pretrained Models Results

The prediction result can be plotted by passing the classification type as a variable to the function plot_result_metrics as shown in the following example. The result shows the performance metrics: Recall, Precision, Balanced Accuracy, HSS, TSS, and Weighted AUC that the model achieves for each time window 12 to 72 hours ahead.

In [5]:
```python
from SEP_utils import plot_result_metrics
#Plotting result for FC_S
plot_result_metrics('FC_S',result_dir='default_results')
```



In [6]:
```python
#Plotting results metrics for F_S
from SEP_utils import plot_result_metrics
plot_result_metrics('F_S',result_dir='default_results')
```



BiLSTM Model Training and Testing Example

BiLSTM Model Training with Sample Data

Here, we show how to train the model with sample data example. In this exmaple, we show how to train the model for tiwe window h = 24 hour for the FC_S classification problems.

In [7]:
```
#Training for FC_S 12-72 hours on sample data.
print('Loading the train_model function...')
from SEP_train import train_model
print('Train model for FC_S classification and time window h=12-72')
classification_type='FC_S'
starting_time_window=12
ending_time_window= 72

train_model(classification_type,starting_time_window,ending_time_window+1)
```

```
Loading the train_model function...
Tensorflow bakcend version: 2.10.0
SUCCESS: Found GPU: /device:GPU:0
Train model for FC_S classification and time window h=12-72


Running classification type: FC_S  training for h = 12 hour ahead
Loading data from data file: data/events_fc_training_12.csv
Loading data from data file: data/events_fc_testing_12.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step

Epoch 3/5
1/1 [==============================] - 3s 3s/step
Epoch 4/5
1/1 [==============================] - 1s 1s/step
Epoch 5/5
1/1 [==============================] - 2s 2s/step



Running classification type: FC_S  training for h = 24 hour ahead
Loading data from data file: data/events_fc_training_24.csv
Loading data from data file: data/events_fc_testing_24.csv
Epoch 1/5
1/1 [==============================] - 2s 2s/step
Epoch 2/5
1/1 [==============================] - 2s 2s/step
Epoch 3/5
1/1 [==============================] - 2s 2s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step



Running classification type: FC_S  training for h = 36 hour ahead
Loading data from data file: data/events_fc_training_36.csv
Loading data from data file: data/events_fc_testing_36.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step

Epoch 3/5
1/1 [==============================] - 1s 1s/step
Epoch 4/5
1/1 [==============================] - 2s 2s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step
```

```
Running classification type: FC_S  training for h = 48 hour ahead
Loading data from data file: data/events_fc_training_48.csv
Loading data from data file: data/events_fc_testing_48.csv
Epoch 1/5
1/1 [==============================] - 2s 2s/step
Epoch 2/5
1/1 [==============================] - 2s 2s/step
Epoch 3/5
1/1 [==============================] - 1s 1s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 2s 2s/step



Running classification type: FC_S  training for h = 60 hour ahead
Loading data from data file: data/events_fc_training_60.csv
Loading data from data file: data/events_fc_testing_60.csv
Epoch 1/5
1/1 [==============================] - 2s 2s/step
Epoch 2/5
1/1 [==============================] - 2s 2s/step
Epoch 3/5
1/1 [==============================] - 1s 1s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step


Running classification type: FC_S  training for h = 72 hour ahead
Loading data from data file: data/events_fc_training_72.csv
Loading data from data file: data/events_fc_testing_72.csv
Epoch 1/5
1/1 [==============================] - 4s 4s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step
Epoch 3/5
1/1 [==============================] - 1s 1s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step
Finished training.
---------------------------------------------
```

In [8]:

```
#Training for FS_S 12-72 hours on sample data.
```

```
print('Loading the train_model function...')
from SEP_train import train_model
print('Train model for F_S classification and time window h=12-72')
classification_type='F_S'
starting_time_window=12
ending_time_window= 72

train_model(classification_type,starting_time_window,ending_time_window+1)
```

```
Loading the train_model function...
Train model for F_S classification and time window h=12-72


Running classification type: F_S  training for h = 12 hour ahead
Loading data from data file: data/events_f_training_12.csv
Loading data from data file: data/events_f_testing_12.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 2s 2s/step

Epoch 3/5
1/1 [==============================] - 1s 1s/step
Epoch 4/5
1/1 [==============================] - 1s 1s/step
Epoch 5/5
1/1 [==============================] - 1s 1s/step



Running classification type: F_S  training for h = 24 hour ahead
Loading data from data file: data/events_f_training_24.csv
Loading data from data file: data/events_f_testing_24.csv
Epoch 1/5
1/1 [==============================] - 2s 2s/step
Epoch 2/5
1/1 [==============================] - 3s 3s/step
Epoch 3/5
1/1 [==============================] - 3s 3s/step

Epoch 4/5
1/1 [==============================] - 4s 4s/step

Epoch 5/5
1/1 [==============================] - 3s 3s/step



Running classification type: F_S  training for h = 36 hour ahead
Loading data from data file: data/events_f_training_36.csv
Loading data from data file: data/events_f_testing_36.csv
Epoch 1/5
1/1 [==============================] - 4s 4s/step
Epoch 2/5
1/1 [==============================] - 4s 4s/step
Epoch 3/5
1/1 [==============================] - 4s 4s/step
Epoch 4/5
1/1 [==============================] - 3s 3s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step


Running classification type: F_S  training for h = 48 hour ahead
Loading data from data file: data/events_f_training_48.csv
```

```
Loading data from data file: data/events_f_testing_48.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step

Epoch 3/5
1/1 [==============================] - 1s 1s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step
Epoch 5/5
1/1 [==============================] - 1s 1s/step



Running classification type: F_S  training for h = 60 hour ahead
Loading data from data file: data/events_f_training_60.csv
Loading data from data file: data/events_f_testing_60.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step
Epoch 3/5
1/1 [==============================] - 1s 1s/step

Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step



Running classification type: F_S  training for h = 72 hour ahead
Loading data from data file: data/events_f_training_72.csv
Loading data from data file: data/events_f_testing_72.csv
Epoch 1/5
1/1 [==============================] - 1s 1s/step
Epoch 2/5
1/1 [==============================] - 1s 1s/step

Epoch 3/5
1/1 [==============================] - 1s 1s/step
Epoch 4/5
1/1 [==============================] - 1s 1s/step

Epoch 5/5
1/1 [==============================] - 1s 1s/step

Finished training.
---------------------------------------------
```

Predicting with Your Trained BiLSTM Model

To predict the testing data using the model you trained above for the FC_S and F_S

classifications, make sure the models_directory variable is set to models:
models_directory='models'

Note: this is training job is only an example that uses less, training proceses, epochs, therefore the results and performance metrics are not comparable to fully developed pretrained and default models.

In [9]:
```
#Test trained model for FC_S and 12-hour
from SEP_test import test

models_directory='models'
print('Test the trained models for first classification type FC_S and for time windows
classification_type='FC_S'
starting_time_window=12
ending_time_window= 72
test(classification_type,starting_time_window,ending_time_window+1,models_directory=mo
```

```
Test the trained models for first classification type FC_S and for time windows h=12.
Running classification test type: FC_S  training for h = 12 hour ahead
testing data file: data/events_fc_testing_12.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_12hr
Building model for: models\sep_model_fc_12hr\model_weights
Loading weights from: models\sep_model_fc_12hr\model_weights
Loading data from data file: data/events_fc_testing_12.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_12.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
12.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 24 hour ahead
testing data file: data/events_fc_testing_24.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_24hr
Building model for: models\sep_model_fc_24hr\model_weights
Loading weights from: models\sep_model_fc_24hr\model_weights
Loading data from data file: data/events_fc_testing_24.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_24.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
24.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 36 hour ahead
testing data file: data/events_fc_testing_36.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_36hr
Building model for: models\sep_model_fc_36hr\model_weights
Loading weights from: models\sep_model_fc_36hr\model_weights
Loading data from data file: data/events_fc_testing_36.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_36.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
36.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 48 hour ahead
testing data file: data/events_fc_testing_48.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_48hr
Building model for: models\sep_model_fc_48hr\model_weights
Loading weights from: models\sep_model_fc_48hr\model_weights
Loading data from data file: data/events_fc_testing_48.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_48.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
48.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 60 hour ahead
testing data file: data/events_fc_testing_60.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_60hr
```

```
Building model for: models\sep_model_fc_60hr\model_weights
Loading weights from: models\sep_model_fc_60hr\model_weights
Loading data from data file: data/events_fc_testing_60.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_60.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
60.csv
-------------------------------------------------------

Running classification test type: FC_S  training for h = 72 hour ahead
testing data file: data/events_fc_testing_72.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_fc_72hr
Building model for: models\sep_model_fc_72hr\model_weights
Loading weights from: models\sep_model_fc_72hr\model_weights
Loading data from data file: data/events_fc_testing_72.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_FC_S_72.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_FC_S_
72.csv
-------------------------------------------------------
```

In [10]:

```python
#Test trained model for F_S and 12-hour
from SEP_test import test

models_directory='models'
print('Test the trained models for second classification type F_S and for time windows
classification_type='F_S'
test(classification_type,starting_time_window,ending_time_window+1,models_directory=mc
```

```
Test the trained models for second classification type F_S and for time windows h=12.
Running classification test type: F_S  training for h = 12 hour ahead
testing data file: data/events_f_testing_12.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_12hr
Building model for: models\sep_model_f_12hr\model_weights
Loading weights from: models\sep_model_f_12hr\model_weights
Loading data from data file: data/events_f_testing_12.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_12.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_1
2.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 24 hour ahead
testing data file: data/events_f_testing_24.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_24hr
Building model for: models\sep_model_f_24hr\model_weights
Loading weights from: models\sep_model_f_24hr\model_weights
Loading data from data file: data/events_f_testing_24.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_24.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_2
4.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 36 hour ahead
testing data file: data/events_f_testing_36.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_36hr
Building model for: models\sep_model_f_36hr\model_weights
Loading weights from: models\sep_model_f_36hr\model_weights
Loading data from data file: data/events_f_testing_36.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_36.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_3
6.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 48 hour ahead
testing data file: data/events_f_testing_48.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_48hr
Building model for: models\sep_model_f_48hr\model_weights
Loading weights from: models\sep_model_f_48hr\model_weights
Loading data from data file: data/events_f_testing_48.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_48.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_4
8.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 60 hour ahead
testing data file: data/events_f_testing_60.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_60hr
```

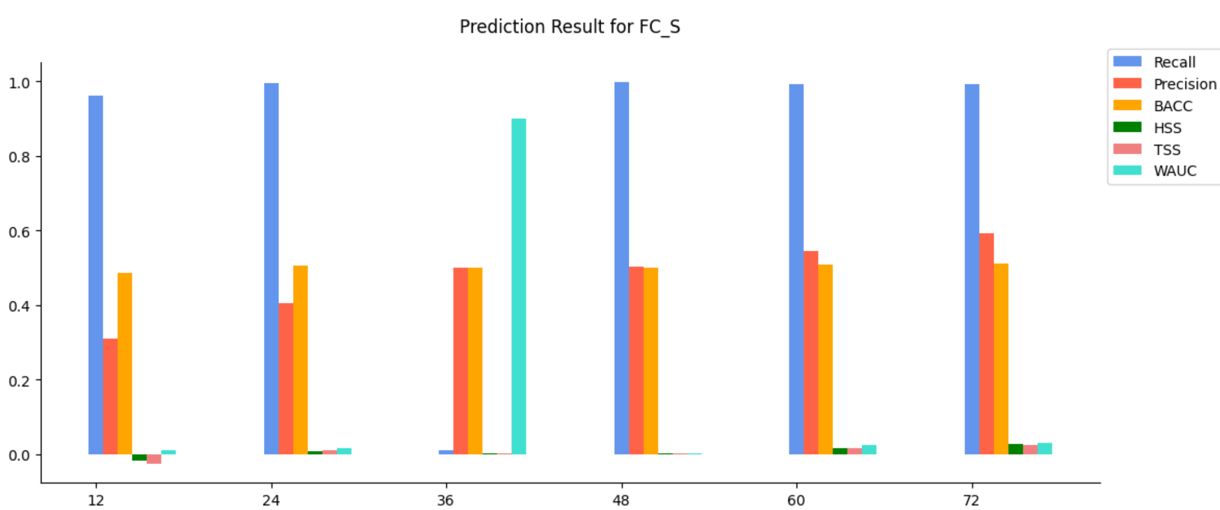
```
Building model for: models\sep_model_f_60hr\model_weights
Loading weights from: models\sep_model_f_60hr\model_weights
Loading data from data file: data/events_f_testing_60.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_60.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_6
0.csv
-------------------------------------------------------

Running classification test type: F_S  training for h = 72 hour ahead
testing data file: data/events_f_testing_72.csv
Loading the model and its weights.
Loading weights from model dir: models\sep_model_f_72hr
Building model for: models\sep_model_f_72hr\model_weights
Loading weights from: models\sep_model_f_72hr\model_weights
Loading data from data file: data/events_f_testing_72.csv
Prediction and calibration..
Saving result to file: results\SEP_prediction_results_F_S_72.csv
Saving the performance metrics to files: results\SEP_performance_metrics_BiLSTM_F_S_7
2.csv
-------------------------------------------------------
```

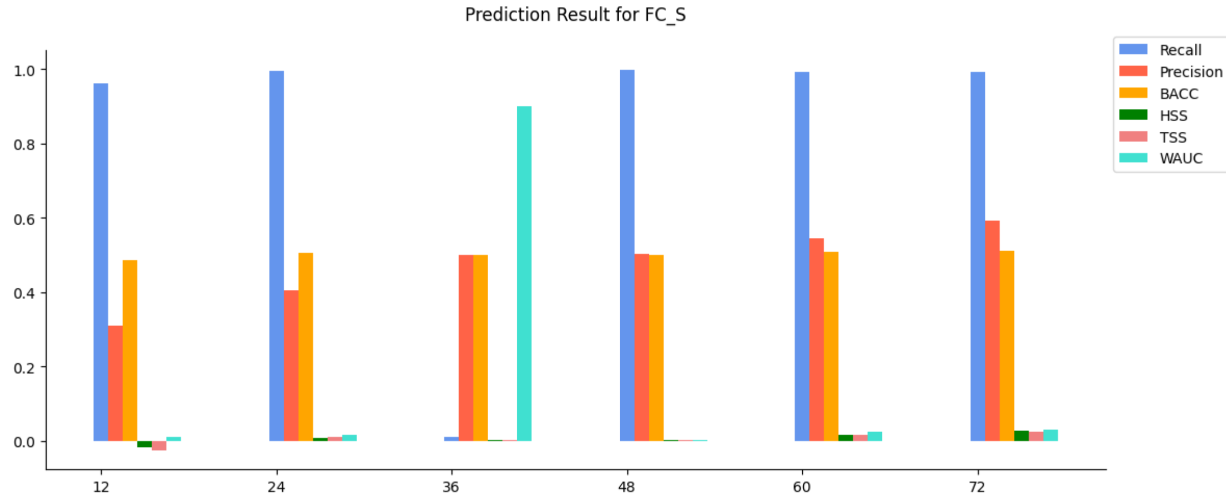
Plotting the Results for Your Trained Model

The prediction result can be plotted by passing the classification type as a variable to the function plot_result_metrics as shown in the following example. You should also pass the time window from the training, ie time_window=12 The result shows the performance metrics: Recall, Precision, Balanced Accuracy, HSS, TSS, and Weighted AUC that the model achieves for each time window 12 to 72 hours ahead.

In [11]:
```python
#Plotting results for trained model for FC_S and time window =12-72
from SEP_utils import plot_result_metrics
plot_result_metrics('FC_S')
```



In [12]:
```python
#Plotting results for trained model for F_S and time window =12-72
from SEP_utils import plot_result_metrics
plot_result_metrics('FC_S')
```



Timing

Please note that the execution time in mybinder varies based on the availability of resources. The average time to run the notebook is 10-15 minutes, but it could be more.

Conclusions

We develop a biLSTM network for SEP prediction. We consider two prediction tasks. In the first task (FC_S), given a data sample $x_t$at time point t in an AR where the AR will produce an M- or X-class flare within the next T hours of t and the flare initiates a CME, based on the SHARP parameters in x_t and its preceding $m-1$ data samples x_{t-m+1}, x t−m+2 , K, x $_{t-1}$, our biLSTM, when used as a binary prediction model, can predict whether the AR will produce an SEP event associated with the flare/CME. Furthermore, our biLSTM, when used as a probabilistic forecasting model, can provide a probabilistic estimate of how likely it is that the AR will produce an SEP event associated with the flare/CME. In the second task (F_S), given a data sample x_t at time point t in an AR where the AR will produce an M- or X-class flare within the next T hours of t, based on the SHARP parameters in x_t and its preceding $m-1$ data samples x_{t-m+1} , x_{t-m+2} , K, x_{t-1} , our biLSTM, when used as a binary prediction model, can predict whether the AR will produce an SEP event associated with the flare, and when used as a probabilistic forecasting model, can provide a probabilistic estimate of how likely it is that the AR will produce an SEP event associated with the flare, regardless of whether or not the flare initiates a CME. For both tasks, T ranges from 12 to 72 in 12 hr intervals.

References

1. Predicting Solar Energetic Particles Using SDO/HMI Vector Magnetic Data Products and a Bidirectional LSTM Network
   Yasser Abduallah, Vania K. Jordanova, Hao Liu, Qin Li, Jason T. L. Wang and Haimin Wang
   https://iopscience.iop.org/article/10.3847/1538-4365/ac5f56

2. DeepSun: Machine-Learning-as-a-Service for Solar Flare Prediction
   Yasser Abduallah, Jason T. L. Wang and Haimin Wang
   https://iopscience.iop.org/article/10.1088/1674-4527/21/7/160
3. Predicting Solar Flares Using SDO/HMI Vector Magnetic Data Products and the Random Forest Algorithm
   Chang Liu, Na Deng, Jason T. L. Wang and Haimin Wang
   https://iopscience.iop.org/article/10.3847/1538-4357/aa789b
4. Artificial Neural Networks: An Introduction to ANN Theory and Practice
   P. J. Braspenning, F. Thuijsman, A. J. M. M. Weijters
   https://link.springer.com/book/10.1007/BFb0027019
5. Using Machine Learning Methods to Forecast if Solar Flares Will Be Associated with CMEs and SEPs
   Fadil Inceoglu, Jacob H. Jeppesen, Peter Kongstad, Nestor J. Hernandez Marcano and Rune H. Jacobsen and Christoffer Karoff
   https://doi.org/10.3847%2F1538-4357%2Faac81e