Deepnote Debmalya › Debmalya Ray’s Untitled project Unlisted Run Debmalya

```
### Internet Firewall Data
```

Additional Information

There are 12 features in total. Action feature is used as a class. There are 4 classes in total. These are allow, action, drop and reset-both classes.

```
###!pip install keras-tuner
```

```
###! unzip /content/internet+firewall+data.zip
```

```
###! pip install tensorflow
###! pip install bayesian-optimization
```

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn import preprocessing
import matplotlib.pyplot as plt
tf.random.set_seed(123)
np.random.seed(123)

# Import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, Dropout
from tensorflow.keras.optimizers import Adam, SGD, RMSprop, Adadelta, Adagrad, Adamax, Nadam, Ftrl
from keras.callbacks import EarlyStopping, ModelCheckpoint
###from keras.wrappers.scikit_learn import KerasClassifier
from math import floor
from sklearn.metrics import make_scorer, accuracy_score
from bayes_opt import BayesianOptimization
from sklearn.model_selection import StratifiedKFold
from keras.layers import LeakyReLU
LeakyReLU = LeakyReLU(alpha=0.1)
import warnings
warnings.filterwarnings('ignore')
pd.set_option("display.max_columns", None)
```

```
log_data = pd.read_csv("/content/log2.csv")
```

```
log_data.columns
```

Run the app to see the outputs

Press the run button in the top right corner

```
log_data.info()
```

Run the app to see the outputs

Press the run button in the top right corner

```
from sklearn.model_selection import train_test_split
```

```
training_data, testing_data = train_test_split(log_data, test_size = 0.3, random_state = 0)
```

```
print(training_data.shape, testing_data.shape)
```

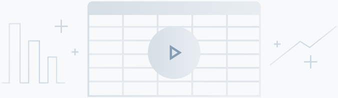
Run the app to see the outputs

Press the run button in the top right corner

```
training_data['ID'] = training_data.index
testing_data['ID'] = testing_data.index
```

```
####! pip install matplotlib_venn
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
```

```
set_numbers_train = set(training_data[['ID']].drop_duplicates().sort_values(by = 'ID')['ID'].tolist())
set_numbers_test = set(testing_data[['ID']].drop_duplicates().sort_values(by = 'ID')['ID'].tolist())
venn2((set_numbers_train, set_numbers_test), set_labels = ('Train numbers', 'Test numbers'))
```



Run the app to see the outputs

Press the run button in the top right corner

```
num_var = [feature for feature in training_data.columns if training_data[feature].dtypes != 'O']
discrete_var = [feature for feature in num_var if len(training_data[feature].unique()) <= 25]
cont_var = [feature for feature in num_var if feature not in discrete_var]
categ_var = [feature for feature in training_data.columns if feature not in num_var]
```

```
###! pip install klib
###!pip install keras-tuner
```

```
import klib
```

```
training_data = klib.clean_column_names(training_data)
testing_data = klib.clean_column_names(testing_data)
```

```
klib.cat_plot(training_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
klib.corr_interactive_plot(training_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
klib.dist_plot(training_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
klib.missingval_plot(training_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
klib.corr_mat(training_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
training_data.columns
```

Run the app to see the outputs

Press the run button in the top right corner

```
training_data['action'].value_counts()
```

Run the app to see the outputs

Press the run button in the top right corner

```
training_data['action'] = training_data['action'].astype('category').cat.codes
testing_data['action'] = testing_data['action'].astype('category').cat.codes
```

```
y_train = training_data['action']
x_train = training_data.drop('action', axis = 1)
y_test = testing_data['action']
x_test = testing_data.drop('action', axis = 1)
```

```
from sklearn.ensemble import ExtraTreesClassifier
extra_tree_forest = ExtraTreesClassifier(n_estimators = 5,
                                        criterion ='entropy', max_features = 2)
extra_tree_forest.fit(x_train, y_train)
feature_importance = extra_tree_forest.feature_importances_
feature_importance_normalized = np.std([tree.feature_importances_ for tree in
                                        extra_tree_forest.estimators_],
                                        axis = 0)
```

```
plt.figure(figsize = [20,5])
plt.bar(x_train.columns, feature_importance_normalized)
plt.xlabel('Feature Labels')
plt.ylabel('Feature Importances')
plt.xticks(rotation = 90)
plt.title('Comparison of different Feature Importances')
plt.show()
```

Run the app to see the outputs

Press the run button in the top right corner

```
print("The columns of the x_train :", x_train.columns)
print("The columns of the x_test :", x_test.columns)
```

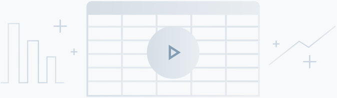
Run the app to see the outputs

Press the run button in the top right corner

```
x_train2 = x_train[['source_port', 'destination_port', 'nat_source_port',
       'nat_destination_port', 'bytes', 'bytes_sent', 'bytes_received',
       'packets', 'elapsed_time_sec', 'pkts_sent', 'pkts_received']]
```

```
x_test2 = x_test[['source_port', 'destination_port', 'nat_source_port',
       'nat_destination_port', 'bytes', 'bytes_sent', 'bytes_received',
       'packets', 'elapsed_time_sec', 'pkts_sent', 'pkts_received']]
```

```
print(x_train2.shape, x_test2.shape)
print(y_train.shape, y_test.shape)
```



Run the app to see the outputs

Press the run button in the top right corner

```
x_train2 = pd.DataFrame(x_train2)
x_test2 = pd.DataFrame(x_test2)
```

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train_scaled=pd.DataFrame(scaler.fit_transform(x_train2),columns=x_train2.columns)
x_train_scaled.head()
```

Run the app to see the outputs

Press the run button in the top right corner

```
x_test_scaled=pd.DataFrame(scaler.fit_transform(x_test2),columns=x_test2.columns)
x_test_scaled.head()
```

Run the app to see the outputs

Press the run button in the top right corner

```
print(x_train_scaled.shape, x_test_scaled.shape)
```

Run the app to see the outputs

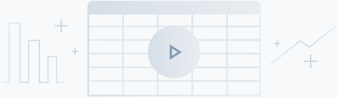
Press the run button in the top right corner

```
y_train = pd.DataFrame(y_train)
y_test = pd.DataFrame(y_test)
```

```
# Install libraries
#!pip install boostaroota
#!pip install h2o
#!pip install ppscore
#!pip install imblearn
```

```
###! pip install optuna
###! pip install shap
```

```
###! pip install catboost
```



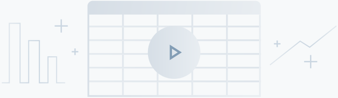
Run the app to see the outputs

Press the run button in the top right corner

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from ipywidgets import interact, interactive, fixed, interact_manual
import ipywidgets as widgets
import plotly.express as px
import matplotlib.pyplot as plt
import plotly.graph_objs as go
from tqdm import tqdm
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from sklearn import model_selection as sk_model_selection
from xgboost.sklearn import XGBRegressor
from sklearn.metrics import mean_squared_error,roc_auc_score,precision_score
from sklearn import metrics
import optuna
from boostaroota import BoostARoota
from sklearn.metrics import log_loss
from optuna.samplers import TPESampler
import functools
from functools import partial
import xgboost as xgb
import joblib
from matplotlib_venn import venn2, venn2_circles, venn2_unweighted
from matplotlib_venn import venn3, venn3_circles
import statsmodels.api as sm
import pylab
from xgboost import plot_tree
import shap
from xgboost.sklearn import XGBClassifier
from sklearn.metrics import mean_squared_error,roc_auc_score,precision_score
from sklearn import metrics
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix, recall_score, precision_score, precision_recall_curve, auc, f1_score, \
    average_precision_score, accuracy_score, roc_curve
from sklearn.preprocessing import LabelEncoder
import h2o
from h2o.automl import H2OAutoML
from catboost import Pool, CatBoostRegressor, cv
import tensorflow as tf
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Dense, Dropout, Input
from tensorflow.keras.layers import Concatenate, LSTM, GRU
from tensorflow.keras.layers import Bidirectional, Multiply
import seaborn as sns
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC


SEED = 42
```

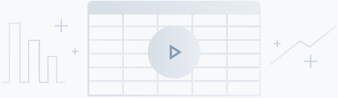
```
print(x_train_scaled.columns)
```



Run the app to see the outputs

Press the run button in the top right corner

```python
print(x_test_scaled.columns)
```



Run the app to see the outputs

Press the run button in the top right corner

```python
feature_cols = ['source_port', 'destination_port', 'nat_source_port',
       'nat_destination_port', 'bytes', 'bytes_sent', 'bytes_received',
       'packets', 'elapsed_time_sec', 'pkts_sent', 'pkts_received']
```

```python
from sklearn.metrics import  precision_recall_curve, roc_auc_score, confusion_matrix, accuracy_score, recall_score, preci
try:
    from imblearn.over_sampling import ADASYN
except:
    pass
try:
    import ppscore as pps
except:
    pass

from imblearn.over_sampling import ADASYN
```

```
h2o.init(
    nthreads=-1,      # number of threads when launching a new H2O server
    max_mem_size=12   # in gigabytes
)
```

Run the app to see the outputs

Press the run button in the top right corner

```
print(training_data.shape, testing_data.shape)
```

Run the app to see the outputs

Press the run button in the top right corner

```
h2o_train_df = h2o.H2OFrame(training_data)
h2o_test_df = h2o.H2OFrame(testing_data)
```

Run the app to see the outputs

Press the run button in the top right corner

```
aml = H2OAutoML(max_models = 35, seed = 100, exclude_algos = ["StackedEnsemble"], verbosity="info", nfolds=0, balance_cla
```

```
training_data.columns
```

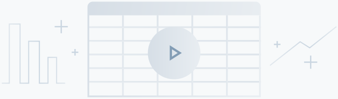
Run the app to see the outputs

Press the run button in the top right corner

```
features = ['source_port', 'destination_port', 'nat_source_port',
       'nat_destination_port', 'bytes', 'bytes_sent',
       'bytes_received', 'packets', 'elapsed_time_sec', 'pkts_sent',
       'pkts_received', 'id']
output = 'action'
```

```
####aml = H2OAutoML(max_models = 30, max_runtime_secs=300, seed = 1)
```

```
aml.train(x = features, y = output, training_frame = h2o_train_df)
```



Run the app to see the outputs

Press the run button in the top right corner

Model leaderboard

```
lb = aml.leaderboard
lb.head(rows=lb.nrows)
```

Run the app to see the outputs

Press the run button in the top right corner

```
preds = aml.predict(h2o_test_df)
```

Run the app to see the outputs

Press the run button in the top right corner

```
preds.columns
```

Run the app to see the outputs

Press the run button in the top right corner

```
preds.head(2)
```

Run the app to see the outputs

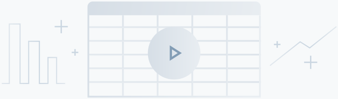
Press the run button in the top right corner

```
sub = preds[0].as_data_frame()
```

```
sub["predict"] = sub.predict.apply(np.round)
```

```
sub["predict"] = sub["predict"].astype(int)
```

```
sub["predict"].value_counts()
```



Run the app to see the outputs

Press the run button in the top right corner