neuron_synapse_prediction

December 12, 2023

Team name on Kaggle: tripleatom

Group Member:

Sheng Cheng. sc159@rice.edu

Xiaorong Zhang. xz106@rice.edu

1 Data preprocessing

1.1 load data

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')

import sys
sys.path.append('drive/Shareddrives/578_term')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
cd /content/drive/Shareddrives/578_term
```

```
/content/drive/Shareddrives/578_term
```

[ ]:
```
!pip install shap
import shap
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.10/dist-packages
(0.44.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from shap) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from shap) (1.11.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from shap) (1.5.3)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-
packages (from shap) (4.66.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-
```

```
packages (from shap) (23.2)
Requirement already satisfied: slicer==0.0.7 in /usr/local/lib/python3.10/dist-
packages (from shap) (0.0.7)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages
(from shap) (0.58.1)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-
packages (from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba->shap) (0.41.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->shap) (2023.3.post1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn->shap) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.1->pandas->shap) (1.16.0)
```

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from imblearn.over_sampling import RandomOverSampler
import warnings
warnings.filterwarnings('ignore')

#load in training data on each potential synapse
data = pd.read_csv("./data/train_data.csv")

#load in additional features for each neuron
feature_weights = pd.read_csv("./data/feature_weights.csv")
morph_embeddings = pd.read_csv("./data/morph_embeddings.csv")

#we need to first load and merge the leaderboard data to have the same format␣
↪as the training set
lb_data = pd.read_csv("./data/leaderboard_data.csv")
```

[ ]:
```
# join all feature_weight_i columns into a single np.array column
feature_weights["feature_weights"] = (
    feature_weights.filter(regex="feature_weight_")
    .sort_index(axis=1)
    .apply(lambda x: np.array(x), axis=1)
)
# delete the feature_weight_i columns
feature_weights.drop(
```

```
    feature_weights.filter(regex="feature_weight_").columns, axis=1,␣
 ↪inplace=True
)

# join all morph_embed_i columns into a single np.array column
morph_embeddings["morph_embeddings"] = (
    morph_embeddings.filter(regex="morph_emb_")
    .sort_index(axis=1)
    .apply(lambda x: np.array(x), axis=1)
)
# delete the morph_embed_i columns
morph_embeddings.drop(
    morph_embeddings.filter(regex="morph_emb_").columns, axis=1, inplace=True
)
```

find neurons without morphology information, and do interpolation

[ ]:
```
# extract pre neucleaus_id columns and convert into np array
pre_id = data['pre_nucleus_id']
pre_id = np.array(pre_id)

morph_id = morph_embeddings['nucleus_id']
morph_id = np.array(morph_id)

# find the indices of the missing morph_id
miss_morph_id  = np.setdiff1d(pre_id, morph_id)

# find the miss id in test data
pre_id_lb = lb_data['pre_nucleus_id']
pre_id_lb = np.array(pre_id_lb)

# find the indices of the missing morph_id
miss_morph_id_test  = np.setdiff1d(pre_id_lb, morph_id)

# merge miss_morph_id and miss_morph_id_test
miss_morph_id = np.union1d(miss_morph_id, miss_morph_id_test)
```

[ ]:
```
# extract the pre_id and pre_neucleus_x, pre_neucleus_y, pre_neucleus_z columns␣
 ↪to a new dataframe
pre_data = data[['pre_nucleus_id', 'pre_nucleus_x', 'pre_nucleus_y',␣
 ↪'pre_nucleus_z']]
# change the name to 'x', 'y', 'z'
pre_data = pre_data.rename(columns={'pre_nucleus_x':'x', 'pre_nucleus_y':'y',␣
 ↪'pre_nucleus_z':'z'})
# change the name to 'nucleus_id'
pre_data = pre_data.rename(columns={'pre_nucleus_id':'nucleus_id'})
```

```
# extract the post_id and post_neucleus_x, post_neucleus_y, post_neucleus_z␣
↪columns to a new dataframe
post_data = data[['post_nucleus_id', 'post_nucleus_x', 'post_nucleus_y',␣
↪'post_nucleus_z']]
post_data = post_data.rename(columns={'post_nucleus_x':'x', 'post_nucleus_y':
↪'y', 'post_nucleus_z':'z'})
post_data = post_data.rename(columns={'post_nucleus_id':'nucleus_id'})

pre_lb_data = lb_data[['pre_nucleus_id', 'pre_nucleus_x', 'pre_nucleus_y',␣
↪'pre_nucleus_z']]
pre_lb_data = pre_lb_data.rename(columns={'pre_nucleus_x':'x', 'pre_nucleus_y':
↪'y', 'pre_nucleus_z':'z'})
pre_lb_data = pre_lb_data.rename(columns={'pre_nucleus_id':'nucleus_id'})

post_lb_data = lb_data[['post_nucleus_id', 'post_nucleus_x', 'post_nucleus_y',␣
↪'post_nucleus_z']]
post_lb_data = post_lb_data.rename(columns={'post_nucleus_x':'x',␣
↪'post_nucleus_y':'y', 'post_nucleus_z':'z'})
post_lb_data = post_lb_data.rename(columns={'post_nucleus_id':'nucleus_id'})

# concat the pre_data, post_data, pre_lb_data, post_lb_data and drop the␣
↪duplicate rows
position_data = pd.concat([pre_data, post_data, pre_lb_data, post_lb_data])
position_data = position_data.drop_duplicates(subset=['nucleus_id'],␣
↪keep='first')

position_data = np.array(position_data)
```

[ ]:
```
# average k nearest neighbor's average to fill the missing morphological data

k = 3

for id in miss_morph_id:
    # find the k nearest neighbors
    id_position = position_data[position_data[:,0] == id]
    id_position = id_position[:,1:4]
    # calculate the distance between id and all other neurons
    distance = np.linalg.norm(position_data[:,1:4] - id_position, axis=1)
    # find the k nearest neighbors index
    k_nearest_index = np.argsort(distance)[1:k+1]
    # extract the k nearest neighbors' id
    k_nearest_id = position_data[k_nearest_index,0]

    # extract the k nearest neighbors' morphological data
    k_nearest_morph = morph_embeddings[morph_embeddings['nucleus_id'].
↪isin(k_nearest_id)]
```

```
    # calculate the average of the k nearest neighbors' morphological data
    k_nearest_morph = np.array(k_nearest_morph['morph_embeddings'])
    k_nearest_morph_mean = np.mean(k_nearest_morph, axis=0)

    # append a new row to the morph_embeddings dataframe, neuron_id = id,␣
↪morphological data = k_nearest_morph
    morph_embeddings = pd.concat([morph_embeddings, pd.DataFrame([[id,␣
↪k_nearest_morph_mean]], columns=['nucleus_id', 'morph_embeddings'])])
```

[ ]:
```
data = (
    data.merge(
        feature_weights.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        feature_weights.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
)
```

2 Feature

2.1 Feature engineering

functional similarity

[ ]:
```
#cosine similarity function
def row_feature_similarity(row):
    pre = row["pre_feature_weights"]
    post = row["post_feature_weights"]
    return (pre * post).sum() / (np.linalg.norm(pre) * np.linalg.norm(post))
```

```
# compute the cosine similarity between the pre- and post- feature weights
data["fw_similarity"] = data.apply(row_feature_similarity, axis=1)
```

structural similarity

[ ]:
```
def row_morph_similarity(row):
    pre = row["pre_morph_embeddings"]
    post = row["post_morph_embeddings"]
    return (pre * post).sum() / (np.linalg.norm(pre) * np.linalg.norm(post))

data["me_similarity"] = data.apply(row_morph_similarity, axis=1)
```

nucleus distance

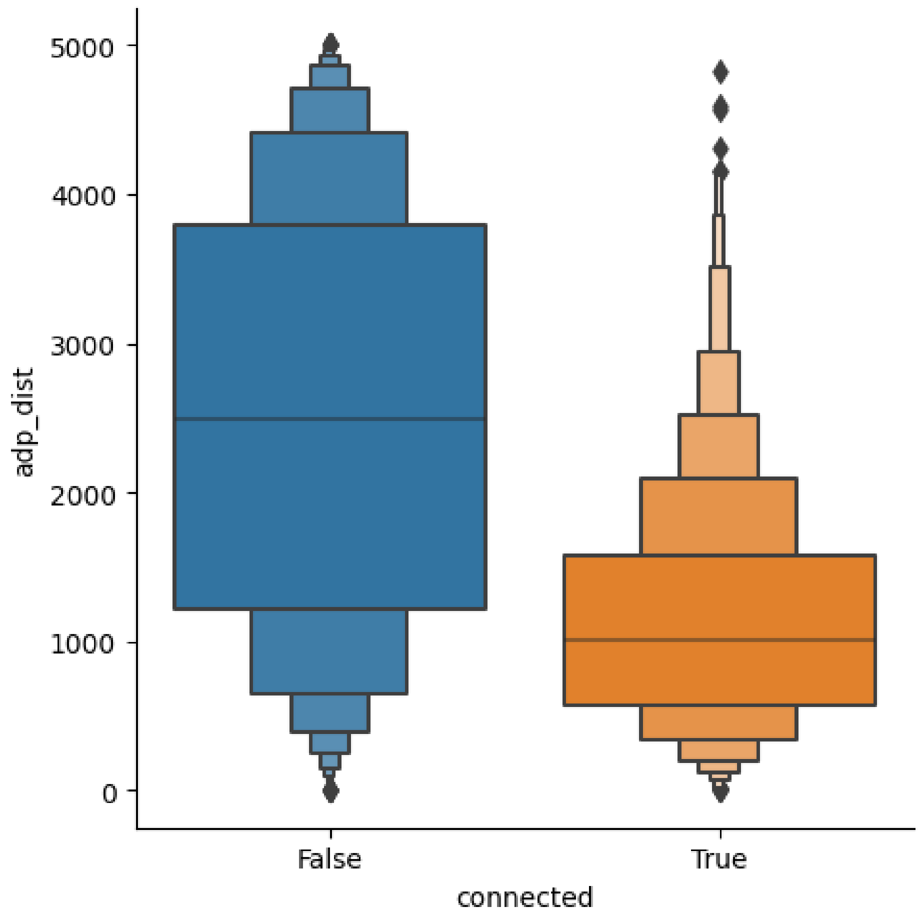
[ ]:
```
data['nucleus_dist'] = np.sqrt((data['pre_nucleus_x'] - 
 ↪data['post_nucleus_x'])**2 + (data['pre_nucleus_y'] - 
 ↪data['post_nucleus_y'])**2 + (data['pre_nucleus_z'] - 
 ↪data['post_nucleus_z'])**2)
# normalize the nucleus_dist
data['nucleus_dist'] = (data['nucleus_dist'] - data['nucleus_dist'].mean()) / 
 ↪data['nucleus_dist'].std()
```

2.2 Feature selection

adp distance

[ ]:
```
sns.catplot(data=data, x='connected', y='adp_dist', kind='boxen')
```

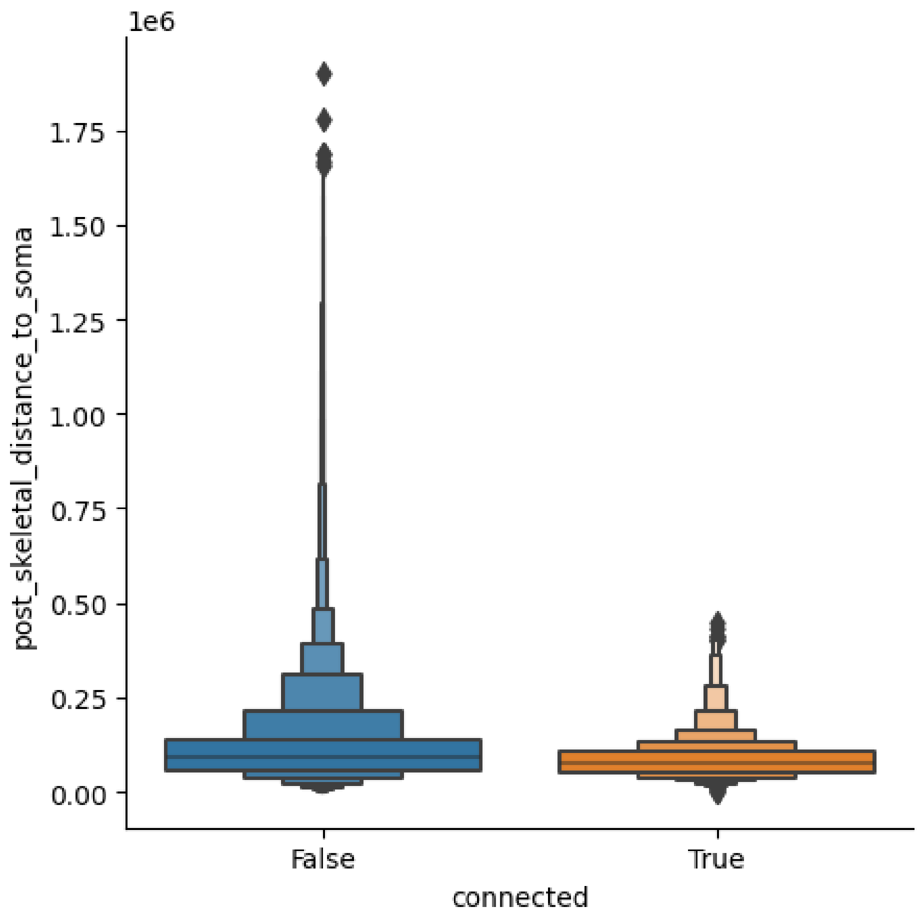
[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7efd37a23160>
```



post skeletal distance to soma

```
[ ]: sns.catplot(data=data, x='connected', y='post_skeletal_distance_to_soma',␣
       ↪kind='boxen')
```

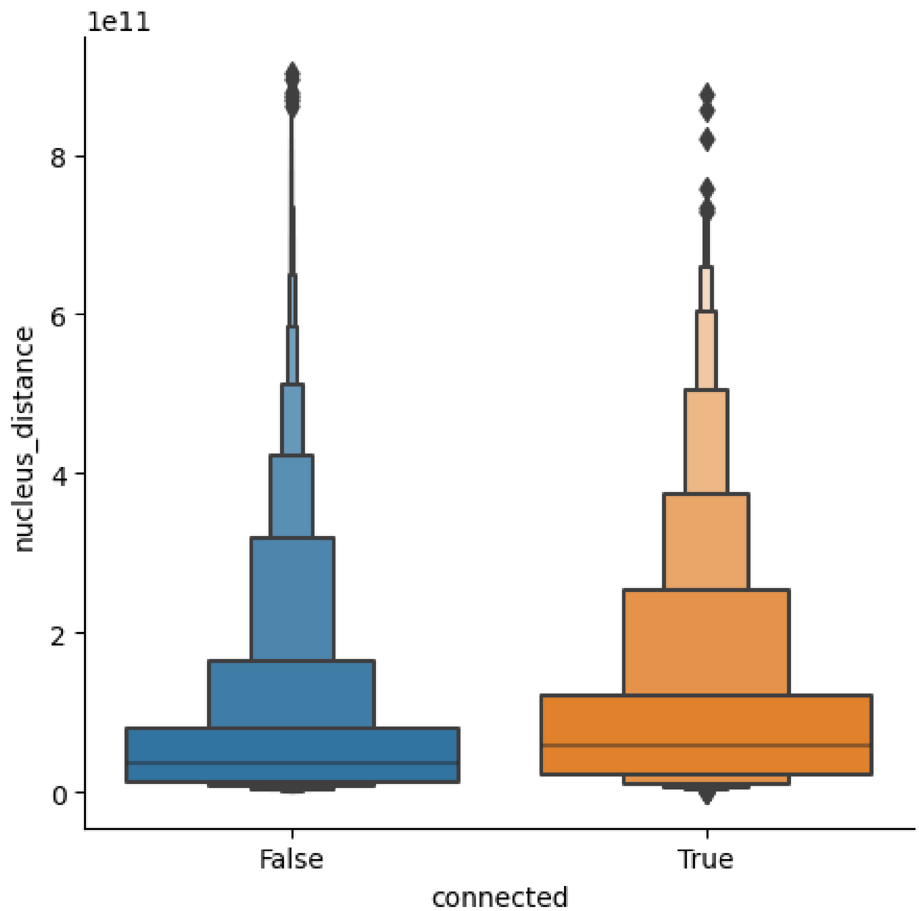
```
[ ]: <seaborn.axisgrid.FacetGrid at 0x7efd3b5b2ec0>
```



nucleus distance

[ ]:
```
# calculate nucleus distance
data["nucleus_distance"] = (
    data["post_nucleus_x"] - data["pre_nucleus_x"]
) ** 2 + (data["post_nucleus_y"] - data["pre_nucleus_y"]) ** 2 + (
    data["post_nucleus_z"] - data["pre_nucleus_z"]
) ** 2

# plot violin plot of nucleus distance
sns.catplot(data=data, x='connected', y='nucleus_distance', kind='boxen')
```

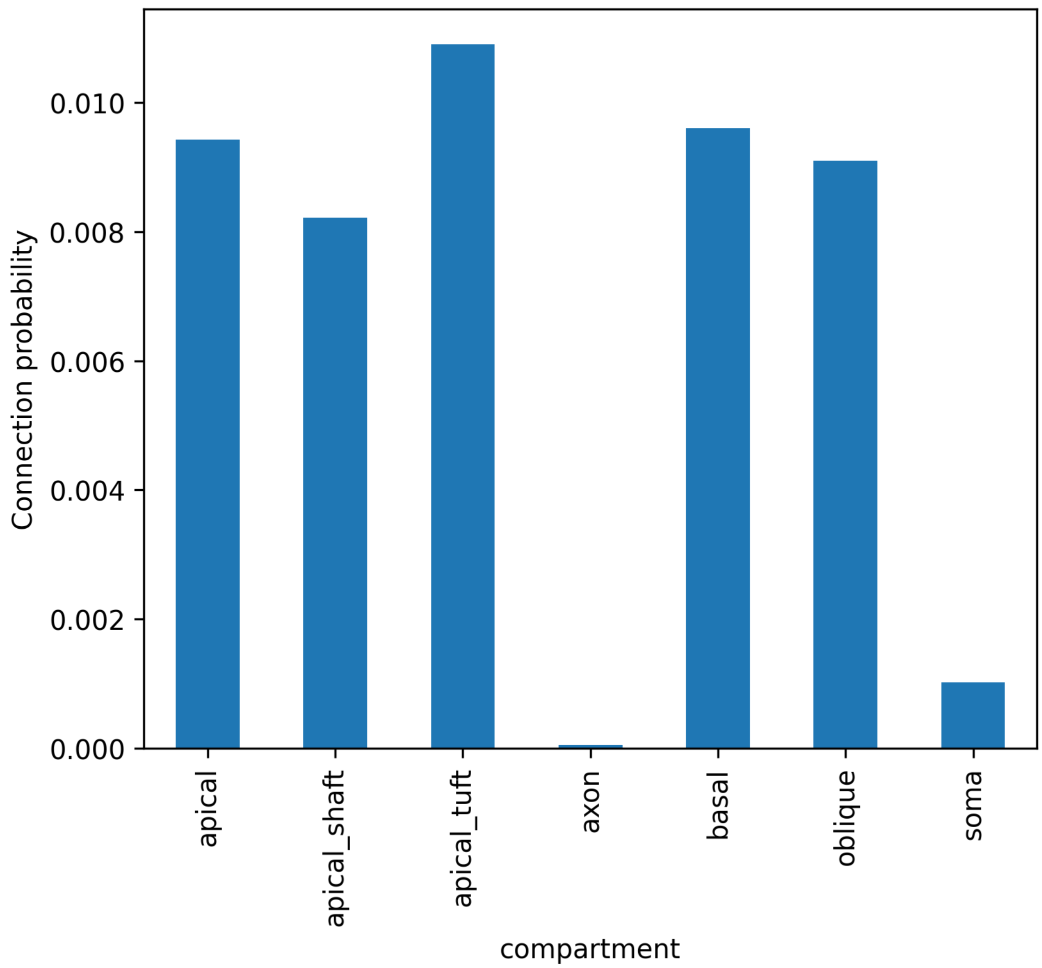
[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7efd30d3bb20>
```



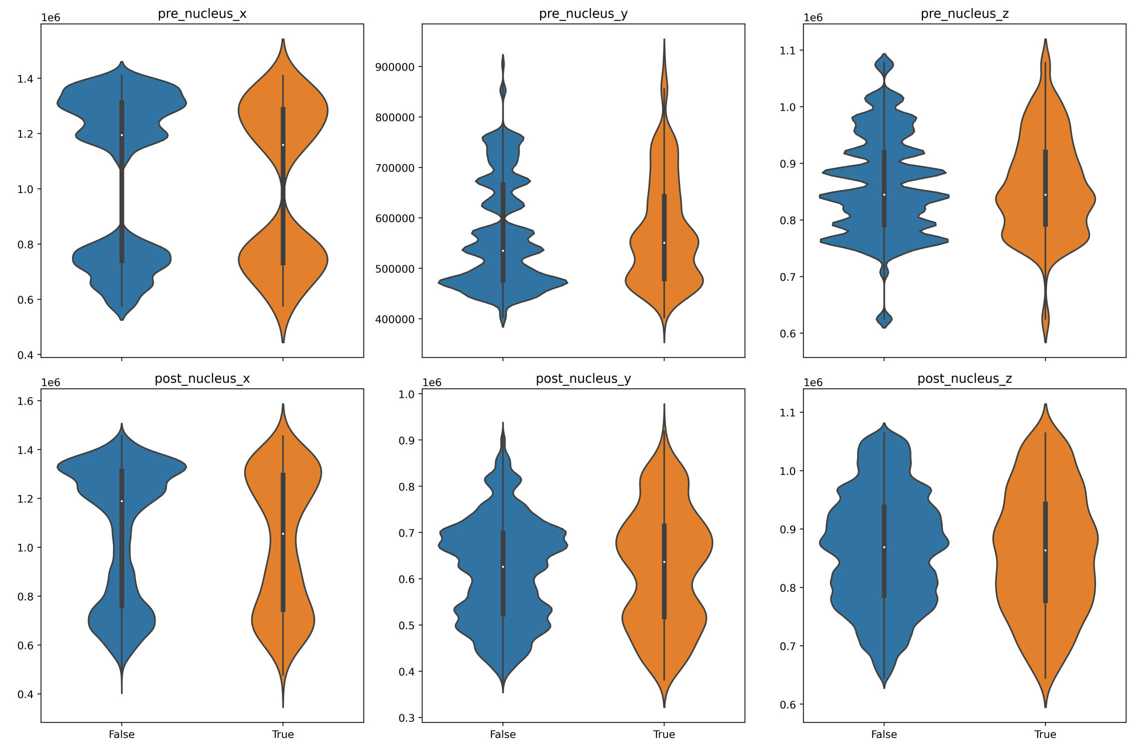
connection propability for differnet compartments of post-synaptic neuron

```
[ ]: # calculate the connection probability for each compartment
     # calculate how many synapses are in each compartment, and how many are␣
      ↪connected
     compartment_counts = data.groupby("compartment")["connected"].value_counts().
      ↪unstack()
     # calculate the connection probability for each compartment
     compartment_counts["connection_probability"] = (
         compartment_counts[True] / compartment_counts.sum(axis=1)
     )

     figure, ax = plt.subplots(figsize=(6, 5), dpi=300)
     # plot the connection probability for each compartment
     compartment_counts["connection_probability"].plot.bar()
     plt.ylabel("Connection probability")
```

```
[ ]: Text(0, 0.5, 'Connection probability')
```




```
[ ]: # plot a 2*3 grid of violin plots of the 6 features, pre_neucleus_x,␣
     ↪pre_neucleus_y, pre_neucleus_z, post_neucleus_x, post_neucleus_y,␣
     ↪post_neucleus_z
     fig, axes = plt.subplots(2, 3, figsize=(15, 10), sharex=True, dpi=400)
     for i, feature in enumerate(["pre_nucleus_x", "pre_nucleus_y", "pre_nucleus_z",␣
     ↪"post_nucleus_x", "post_nucleus_y", "post_nucleus_z"]):
         sns.violinplot(data=data, x='connected', y=feature, ax=axes[i//3][i%3])
         axes[i//3][i%3].set_ylabel("")
         axes[i//3][i%3].set_xlabel("")
         axes[i//3][i%3].set_title(feature)
     plt.tight_layout()
```




```
[ ]: drop_columns=['pre_feature_weights', 'post_feature_weights', 'pre_morph_embeddings', 'post_morph_embeddings', 'pre_nucleus_id', 'post_nucleus_id', 'ID']

remain_columns=['adp_dist','post_skeletal_distance_to_soma', 'pre_skeletal_distance_to_soma','fw_similarity', 'me_similarity', 'post_nucleus_x', 'nucleus_dist']

relation_onehot_columns=[
    ['pre_brain_area', 'post_brain_area'], # 1. use xor to get new feature.  2. no, found out connected neurons can located in both.
                        ]

relation_number_columns=[
    ['pre_nucleus_x','pre_nucleus_y','pre_nucleus_z', 'post_nucleus_y', 'post_nucleus_z'], # get the distance between pre and post
    ['pre_rf_x', 'pre_rf_y', 'post_rf_x', 'post_rf_y'],
]

question_columns=['pre_oracle', 'pre_test_score', 'post_oracle', 'post_test_score'] # oracle relates to neuron reliability, test_score relates to similarity.
```

```
target_columns=['connected']

remain_columns.extend(question_columns)
data2=data[remain_columns].copy(deep=True)

data2['compartment']=data['compartment'].astype("category")
data2['target']=data['connected']
```

[ ]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 185832 entries, 0 to 185831
Data columns (total 13 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   adp_dist                      185832 non-null  float64
 1   post_skeletal_distance_to_soma  185832 non-null  float64
 2   pre_skeletal_distance_to_soma  185832 non-null  float64
 3   fw_similarity                 185832 non-null  float64
 4   me_similarity                 185832 non-null  float64
 5   post_nucleus_x                185832 non-null  int64
 6   pre_oracle                    185832 non-null  float64
 7   pre_test_score                185832 non-null  float64
 8   post_oracle                   185832 non-null  float64
 9   post_test_score               185832 non-null  float64
 10  nucleus_dist                  185832 non-null  float64
 11  compartment                   185832 non-null  category
 12  target                        185832 non-null  bool
dtypes: bool(1), category(1), float64(10), int64(1)
memory usage: 17.4 MB
```

3 LGBM

3.1 downsample and weak classifier

[ ]:
```
!pip install lightgbm==3.3.3
```

```
Requirement already satisfied: lightgbm==3.3.3 in
/usr/local/lib/python3.10/dist-packages (3.3.3)
Requirement already satisfied: wheel in /usr/local/lib/python3.10/dist-packages
(from lightgbm==3.3.3) (0.42.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from lightgbm==3.3.3) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from lightgbm==3.3.3) (1.11.4)
Requirement already satisfied: scikit-learn!=0.22.0 in
/usr/local/lib/python3.10/dist-packages (from lightgbm==3.3.3) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from scikit-learn!=0.22.0->lightgbm==3.3.3) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0->lightgbm==3.3.3) (3.2.0)
```

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import balanced_accuracy_score, accuracy_score,␣
 ↪confusion_matrix
import lightgbm as lgb

x=data2.drop(columns=['target'])
y=data2['target']*1

ratio=y.value_counts()[0]/y.value_counts()[1]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)

#######################################################
# get best hyper parameters with 0.8 training data
#######################################################
# sample negatives, to make balanced dataset
np.random.seed(42)
def eval_balanced_accuracy_score(labels, preds):
    is_higher_better = True
    preds=preds>0.5
    score=balanced_accuracy_score(labels, preds)
    return "bacc", score, is_higher_better

data3=data2.copy(deep=True)

# generate 100 balanced datasets and get 100 models
models=[0]*100
best_lrs=[0]*100
for sample_ix in range(100):
    print(f"**************{sample_ix}****************")
    # data
    data3_neg=data3[data3['target']==False].sample(n=1366)
    data3_pos=data3[data3['target']==True]
    data3=pd.concat([data3_pos,data3_neg])

    data3['target'].value_counts()

    x=data3.drop(columns=['target'])
    y=data3['target']*1

    ratio=y.value_counts()[0]/y.value_counts()[1]
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42, stratify=y)

#model
best_acc=-1
for lr in [1e-2, 1e-3, 1e-4, 1e-5]:
    print(f"-----{lr}------")
    fit_params = {"early_stopping_rounds": 10,
                  "eval_metric": eval_balanced_accuracy_score,
                  "eval_set": [(x_test, y_test)],
                  'eval_names': ['valid'],
                  'verbose': 100,
                  'feature_name': 'auto',  # that's actually the default
                  'categorical_feature': 'auto'  # that's actually the␣
↪default
                }
    model = lgb.LGBMClassifier(
                              scale_pos_weight=ratio,
                              num_leaves=100, max_depth=20,
                              random_state=314,
                              silent=True,
                              n_estimators=2000,
                              colsample_bytree=0.8,
                              subsample=0.9,
                              learning_rate=lr,
                              objective="binary"
                              )
    model.fit(x_train, y_train, **fit_params)

    # predict on test data
    pred_test = model.predict(x_test)

    # compute accuracy
    print(f"accuracy: {accuracy_score(y_test, pred_test)}")

    # confusion matrix
    print(confusion_matrix(y_test, pred_test))

    # compute balanced accuracy
    acc=balanced_accuracy_score(y_test, pred_test)
    print(
        f"balanced accuracy: {acc}"
    )

    if acc>best_acc:
```

```
            best_acc=acc
            models[sample_ix]=model
            best_lrs[sample_ix]=lr
```

```
*************0****************
-----0.01------
accuracy: 0.7842778793418648
[[189  85]
 [ 33 240]]
balanced accuracy: 0.7844509505093447
-----0.001------
accuracy: 0.7824497257769653
[[187  87]
 [ 32 241]]
balanced accuracy: 0.7826328173043502
-----0.0001------
accuracy: 0.7824497257769653
[[172 102]
 [ 17 256]]
balanced accuracy: 0.7827330820031551
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************1****************
-----0.01------
accuracy: 0.7714808043875686
[[183  91]
 [ 34 239]]
balanced accuracy: 0.7716705435683537
-----0.001------
accuracy: 0.7586837294332724
[[181  93]
 [ 39 234]]
balanced accuracy: 0.7588633993743483
-----0.0001------
accuracy: 0.7696526508226691
[[168 106]
 [ 20 253]]
balanced accuracy: 0.7699393064356568
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************2****************
-----0.01------
```

```
accuracy: 0.7769652650822669
[[196  78]
 [ 44 229]]
balanced accuracy: 0.7770781529905617
-----0.001------
accuracy: 0.7751371115173674
[[187  87]
 [ 36 237]]
balanced accuracy: 0.7753068099783429
-----0.0001------
accuracy: 0.7696526508226691
[[169 105]
 [ 21 252]]
balanced accuracy: 0.7699326221224032
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************3*****************
-----0.01------
accuracy: 0.7678244972577697
[[199  75]
 [ 52 221]]
balanced accuracy: 0.7679005908932917
-----0.001------
accuracy: 0.7696526508226691
[[184  90]
 [ 36 237]]
balanced accuracy: 0.7698323574235983
-----0.0001------
accuracy: 0.7751371115173674
[[176  98]
 [ 25 248]]
balanced accuracy: 0.775380337424133
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************4*****************
-----0.01------
accuracy: 0.7586837294332724
[[188  86]
 [ 46 227]]
balanced accuracy: 0.7588166091815727
-----0.001------
accuracy: 0.7605118829981719
```

```
[[178  96]
 [ 35 238]]
balanced accuracy: 0.7607149541456111
-----0.0001------
accuracy: 0.7769652650822669
[[176  98]
 [ 24 249]]
balanced accuracy: 0.7772118392556349
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************5*****************
-----0.01------
accuracy: 0.7586837294332724
[[190  84]
 [ 48 225]]
balanced accuracy: 0.7588032405550653
-----0.001------
accuracy: 0.773308957952468
[[183  91]
 [ 33 240]]
balanced accuracy: 0.7735020453998556
-----0.0001------
accuracy: 0.7605118829981719
[[163 111]
 [ 20 253]]
balanced accuracy: 0.7608152188444159
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************6*****************
-----0.01------
accuracy: 0.7678244972577697
[[207  67]
 [ 60 213]]
balanced accuracy: 0.7678471163872624
-----0.001------
accuracy: 0.7769652650822669
[[188  86]
 [ 36 237]]
balanced accuracy: 0.777131627496591
-----0.0001------
accuracy: 0.7769652650822669
[[170 104]
```

```
 [ 18 255]]
balanced accuracy: 0.7772519451351568
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************7*****************
-----0.01------
accuracy: 0.7879341864716636
[[199  75]
 [ 41 232]]
balanced accuracy: 0.7880471110398117
-----0.001------
accuracy: 0.7879341864716636
[[195  79]
 [ 37 236]]
balanced accuracy: 0.7880738482928265
-----0.0001------
accuracy: 0.7861060329067642
[[178  96]
 [ 21 252]]
balanced accuracy: 0.7863559797866367
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************8*****************
-----0.01------
accuracy: 0.7952468007312614
[[199  75]
 [ 37 236]]
balanced accuracy: 0.795373118365819
-----0.001------
accuracy: 0.7897623400365631
[[197  77]
 [ 38 235]]
balanced accuracy: 0.7898919814978209
-----0.0001------
accuracy: 0.7897623400365631
[[186  88]
 [ 27 246]]
balanced accuracy: 0.7899655089436111
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
*************9****************
-----0.01------
accuracy: 0.7641681901279708
[[188  86]
 [ 43 230]]
balanced accuracy: 0.7643111146760782
-----0.001------
accuracy: 0.7659963436928702
[[176  98]
 [ 30 243]]
balanced accuracy: 0.7662228282666239
-----0.0001------
accuracy: 0.7787934186471663
[[164 110]
 [ 11 262]]
balanced accuracy: 0.7791235528461806
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************10****************
-----0.01------
accuracy: 0.773308957952468
[[188  86]
 [ 38 235]]
balanced accuracy: 0.7734686238335874
-----0.001------
accuracy: 0.7659963436928702
[[184  90]
 [ 38 235]]
balanced accuracy: 0.7661693537605947
-----0.0001------
accuracy: 0.7659963436928702
[[163 111]
 [ 17 256]]
balanced accuracy: 0.7663097243389214
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************11****************
-----0.01------
accuracy: 0.7714808043875686
[[199  75]
 [ 50 223]]
```

```
balanced accuracy: 0.7715635945562953
-----0.001------
accuracy: 0.7714808043875686
[[193  81]
 [ 44 229]]
balanced accuracy: 0.7716037004358172
-----0.0001------
accuracy: 0.7568555758683729
[[166 108]
 [ 25 248]]
balanced accuracy: 0.7571321622416514
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************12*****************
-----0.01------
accuracy: 0.7641681901279708
[[180  94]
 [ 35 238]]
balanced accuracy: 0.7643645891821074
-----0.001------
accuracy: 0.7623400365630713
[[180  94]
 [ 36 237]]
balanced accuracy: 0.7625330873506055
-----0.0001------
accuracy: 0.7678244972577697
[[163 111]
 [ 16 257]]
balanced accuracy: 0.7681412261704232
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************13*****************
-----0.01------
accuracy: 0.7806215722120659
[[195  79]
 [ 41 232]]
balanced accuracy: 0.780747840966819
-----0.001------
accuracy: 0.793418647166362
[[196  78]
 [ 35 238]]
balanced accuracy: 0.7935616694740782
```

```
-----0.0001------
accuracy: 0.7879341864716636
[[183  91]
 [ 25 248]]
balanced accuracy: 0.7881540600518703
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************14*****************
-----0.01------
accuracy: 0.7550274223034735
[[191  83]
 [ 51 222]]
balanced accuracy: 0.7551335525788081
-----0.001------
accuracy: 0.7495429616087751
[[172 102]
 [ 35 238]]
balanced accuracy: 0.749766049036122
-----0.0001------
accuracy: 0.7495429616087751
[[163 111]
 [ 26 247]]
balanced accuracy: 0.749826207855405
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************15*****************
-----0.01------
accuracy: 0.7586837294332724
[[176  98]
 [ 34 239]]
balanced accuracy: 0.7588968209406166
-----0.001------
accuracy: 0.7385740402193784
[[165 109]
 [ 34 239]]
balanced accuracy: 0.7388238282398867
-----0.0001------
accuracy: 0.7659963436928702
[[168 106]
 [ 22 251]]
balanced accuracy: 0.7662763027726531
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************16*****************
-----0.01------
accuracy: 0.7586837294332724
[[183  91]
 [ 41 232]]
balanced accuracy: 0.758850030747841
-----0.001------
accuracy: 0.7586837294332724
[[180  94]
 [ 38 235]]
balanced accuracy: 0.7588700836876019
-----0.0001------
accuracy: 0.7641681901279708
[[165 109]
 [ 20 253]]
balanced accuracy: 0.7644648538809122
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************17*****************
-----0.01------
accuracy: 0.7751371115173674
[[186  88]
 [ 35 238]]
balanced accuracy: 0.7753134942915965
-----0.001------
accuracy: 0.7678244972577697
[[180  94]
 [ 33 240]]
balanced accuracy: 0.768027592845111
-----0.0001------
accuracy: 0.7806215722120659
[[173 101]
 [ 19 254]]
balanced accuracy: 0.7808948958583994
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************18*****************
-----0.01------
```

```
accuracy: 0.7678244972577697
[[194  80]
 [ 47 226]]
balanced accuracy: 0.76793401245956
-----0.001------
accuracy: 0.7605118829981719
[[178  96]
 [ 35 238]]
balanced accuracy: 0.7607149541456111
-----0.0001------
accuracy: 0.7623400365630713
[[165 109]
 [ 21 252]]
balanced accuracy: 0.7626333520494104
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************19*****************
-----0.01------
accuracy: 0.7751371115173674
[[187  87]
 [ 36 237]]
balanced accuracy: 0.7753068099783429
-----0.001------
accuracy: 0.7623400365630713
[[185  89]
 [ 41 232]]
balanced accuracy: 0.7624996657843373
-----0.0001------
accuracy: 0.7714808043875686
[[173 101]
 [ 24 249]]
balanced accuracy: 0.7717373867008903
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************20*****************
-----0.01------
accuracy: 0.7714808043875686
[[186  88]
 [ 37 236]]
balanced accuracy: 0.7716504906285928
-----0.001------
accuracy: 0.7751371115173674
```

```
[[186  88]
 [ 35 238]]
balanced accuracy: 0.7753134942915965
-----0.0001------
accuracy: 0.7678244972577697
[[170 104]
 [ 23 250]]
balanced accuracy: 0.7680944359776476
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************21****************
-----0.01------
accuracy: 0.7714808043875686
[[188  86]
 [ 39 234]]
balanced accuracy: 0.7716371220020855
-----0.001------
accuracy: 0.7623400365630713
[[191  83]
 [ 47 226]]
balanced accuracy: 0.7624595599048154
-----0.0001------
accuracy: 0.773308957952468
[[165 109]
 [ 15 258]]
balanced accuracy: 0.7736223630384214
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************22****************
-----0.01------
accuracy: 0.7751371115173674
[[202  72]
 [ 51 222]]
balanced accuracy: 0.775206545279538
-----0.001------
accuracy: 0.773308957952468
[[190  84]
 [ 40 233]]
balanced accuracy: 0.77345525520708
-----0.0001------
accuracy: 0.7751371115173674
[[169 105]
```

```
 [ 18 255]]
balanced accuracy: 0.7754271276169087
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************23****************
-----0.01------
accuracy: 0.7458866544789763
[[191  83]
 [ 56 217]]
balanced accuracy: 0.7459760434212989
-----0.001------
accuracy: 0.7513711151736746
[[184  90]
 [ 46 227]]
balanced accuracy: 0.75151733910858
-----0.0001------
accuracy: 0.7623400365630713
[[172 102]
 [ 28 245]]
balanced accuracy: 0.7625865618566349
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************24****************
-----0.01------
accuracy: 0.7641681901279708
[[183  91]
 [ 38 235]]
balanced accuracy: 0.7643445362423464
-----0.001------
accuracy: 0.7568555758683729
[[178  96]
 [ 37 236]]
balanced accuracy: 0.7570519504826074
-----0.0001------
accuracy: 0.7678244972577697
[[168 106]
 [ 21 252]]
balanced accuracy: 0.768107804604155
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
*************25****************
-----0.01------
accuracy: 0.7605118829981719
[[185  89]
 [ 42 231]]
balanced accuracy: 0.7606681639528354
-----0.001------
accuracy: 0.7605118829981719
[[180  94]
 [ 37 236]]
balanced accuracy: 0.7607015855191037
-----0.0001------
accuracy: 0.7586837294332724
[[168 106]
 [ 26 247]]
balanced accuracy: 0.7589502954466458
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************26****************
-----0.01------
accuracy: 0.7861060329067642
[[199  75]
 [ 42 231]]
balanced accuracy: 0.7862156092083099
-----0.001------
accuracy: 0.7806215722120659
[[196  78]
 [ 42 231]]
balanced accuracy: 0.7807411566535654
-----0.0001------
accuracy: 0.7623400365630713
[[164 110]
 [ 20 253]]
balanced accuracy: 0.7626400363626642
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************27****************
-----0.01------
accuracy: 0.7623400365630713
[[187  87]
 [ 43 230]]
```

```
balanced accuracy: 0.76248629715783
-----0.001------
accuracy: 0.7568555758683729
[[172 102]
 [ 31 242]]
balanced accuracy: 0.7570920563621293
-----0.0001------
accuracy: 0.7586837294332724
[[155 119]
 [ 13 260]]
balanced accuracy: 0.7590371915189433
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************28****************
-----0.01------
accuracy: 0.7714808043875686
[[187  87]
 [ 38 235]]
balanced accuracy: 0.7716438063153392
-----0.001------
accuracy: 0.773308957952468
[[189  85]
 [ 39 234]]
balanced accuracy: 0.7734619395203337
-----0.0001------
accuracy: 0.7787934186471663
[[174 100]
 [ 21 252]]
balanced accuracy: 0.779056709713644
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************29****************
-----0.01------
accuracy: 0.7641681901279708
[[192  82]
 [ 47 226]]
balanced accuracy: 0.7642843774230635
-----0.001------
accuracy: 0.7751371115173674
[[183  91]
 [ 32 241]]
balanced accuracy: 0.7753335472313574
```

```
-----0.0001------
accuracy: 0.7641681901279708
[[172 102]
 [ 27 246]]
balanced accuracy: 0.7644180636881367
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************30*****************
-----0.01------
accuracy: 0.7769652650822669
[[189  85]
 [ 37 236]]
balanced accuracy: 0.7771249431833374
-----0.001------
accuracy: 0.7769652650822669
[[184  90]
 [ 32 241]]
balanced accuracy: 0.7771583647496056
-----0.0001------
accuracy: 0.7659963436928702
[[167 107]
 [ 21 252]]
balanced accuracy: 0.7662829870859068
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************31*****************
-----0.01------
accuracy: 0.7842778793418648
[[204  70]
 [ 48 225]]
balanced accuracy: 0.7843506858105398
-----0.001------
accuracy: 0.7915904936014625
[[190  84]
 [ 30 243]]
balanced accuracy: 0.7917702735220984
-----0.0001------
accuracy: 0.7861060329067642
[[175  99]
 [ 18 255]]
balanced accuracy: 0.7863760327263978
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************32****************
-----0.01------
accuracy: 0.753199268738574
[[195  79]
 [ 56 217]]
balanced accuracy: 0.7532753134942916
-----0.001------
accuracy: 0.7385740402193784
[[164 110]
 [ 33 240]]
balanced accuracy: 0.7388305125531403
-----0.0001------
accuracy: 0.7568555758683729
[[162 112]
 [ 21 252]]
balanced accuracy: 0.757158899494666
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************33****************
-----0.01------
accuracy: 0.7879341864716636
[[196  78]
 [ 38 235]]
balanced accuracy: 0.7880671639795728
-----0.001------
accuracy: 0.7696526508226691
[[193  81]
 [ 45 228]]
balanced accuracy: 0.7697721986043153
-----0.0001------
accuracy: 0.773308957952468
[[172 102]
 [ 22 251]]
balanced accuracy: 0.7735755728456458
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************34****************
-----0.01------
```

```
accuracy: 0.7678244972577697
[[184  90]
 [ 37 236]]
balanced accuracy: 0.7680008555920965
-----0.001------
accuracy: 0.7477148080438757
[[171 103]
 [ 35 238]]
balanced accuracy: 0.7479412315178738
-----0.0001------
accuracy: 0.7696526508226691
[[162 112]
 [ 14 259]]
balanced accuracy: 0.7699794123151787
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************35****************
-----0.01------
accuracy: 0.7605118829981719
[[186  88]
 [ 43 230]]
balanced accuracy: 0.7606614796395819
-----0.001------
accuracy: 0.7477148080438757
[[178  96]
 [ 42 231]]
balanced accuracy: 0.7478944413250983
-----0.0001------
accuracy: 0.7568555758683729
[[160 114]
 [ 19 254]]
balanced accuracy: 0.7571722681211732
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************36****************
-----0.01------
accuracy: 0.7568555758683729
[[175  99]
 [ 34 239]]
balanced accuracy: 0.7570720034223684
-----0.001------
accuracy: 0.753199268738574
```

```
[[175  99]
 [ 36 237]]
balanced accuracy: 0.7534089997593647
-----0.0001------
accuracy: 0.7550274223034735
[[161 113]
 [ 21 252]]
balanced accuracy: 0.7553340819764178
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************37****************
-----0.01------
accuracy: 0.7751371115173674
[[185  89]
 [ 34 239]]
balanced accuracy: 0.7753201786048501
-----0.001------
accuracy: 0.7568555758683729
[[180  94]
 [ 39 234]]
balanced accuracy: 0.7570385818561001
-----0.0001------
accuracy: 0.7678244972577697
[[164 110]
 [ 17 256]]
balanced accuracy: 0.7681345418571697
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************38****************
-----0.01------
accuracy: 0.7623400365630713
[[184  90]
 [ 40 233]]
balanced accuracy: 0.7625063500975909
-----0.001------
accuracy: 0.7495429616087751
[[172 102]
 [ 35 238]]
balanced accuracy: 0.749766049036122
-----0.0001------
accuracy: 0.753199268738574
[[160 114]
```

```
 [ 21 252]]
balanced accuracy: 0.7535092644581696
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************39*****************
-----0.01------
accuracy: 0.7696526508226691
[[180  94]
 [ 32 241]]
balanced accuracy: 0.7698590946766128
-----0.001------
accuracy: 0.7586837294332724
[[175  99]
 [ 33 240]]
balanced accuracy: 0.7589035052538702
-----0.0001------
accuracy: 0.7568555758683729
[[165 109]
 [ 24 249]]
balanced accuracy: 0.7571388465549049
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************40*****************
-----0.01------
accuracy: 0.7714808043875686
[[184  90]
 [ 35 238]]
balanced accuracy: 0.7716638592551002
-----0.001------
accuracy: 0.7623400365630713
[[172 102]
 [ 28 245]]
balanced accuracy: 0.7625865618566349
-----0.0001------
accuracy: 0.7678244972577697
[[169 105]
 [ 22 251]]
balanced accuracy: 0.7681011202909013
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
**************41*****************
-----0.01------
accuracy: 0.7751371115173674
[[193  81]
 [ 42 231]]
balanced accuracy: 0.7752667040988208
-----0.001------
accuracy: 0.7787934186471663
[[191  83]
 [ 38 235]]
balanced accuracy: 0.7789430763883318
-----0.0001------
accuracy: 0.7824497257769653
[[180  94]
 [ 25 248]]
balanced accuracy: 0.7826796074971257
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************42*****************
-----0.01------
accuracy: 0.773308957952468
[[196  78]
 [ 46 227]]
balanced accuracy: 0.7734151493275581
-----0.001------
accuracy: 0.7641681901279708
[[175  99]
 [ 30 243]]
balanced accuracy: 0.7643980107483757
-----0.0001------
accuracy: 0.7641681901279708
[[164 110]
 [ 19 254]]
balanced accuracy: 0.764471538194166
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************43*****************
-----0.01------
accuracy: 0.7787934186471663
[[194  80]
 [ 41 232]]
```

```
balanced accuracy: 0.7789230234485709
-----0.001------
accuracy: 0.7714808043875686
[[177  97]
 [ 28 245]]
balanced accuracy: 0.7717106494478758
-----0.0001------
accuracy: 0.7678244972577697
[[154 120]
 [  7 266]]
balanced accuracy: 0.7682013849897061
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************44****************
-----0.01------
accuracy: 0.773308957952468
[[191  83]
 [ 41 232]]
balanced accuracy: 0.7734485708938263
-----0.001------
accuracy: 0.7769652650822669
[[188  86]
 [ 36 237]]
balanced accuracy: 0.777131627496591
-----0.0001------
accuracy: 0.7623400365630713
[[154 120]
 [ 10 263]]
balanced accuracy: 0.7627068794952007
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************45****************
-----0.01------
accuracy: 0.7586837294332724
[[191  83]
 [ 49 224]]
balanced accuracy: 0.7587965562418117
-----0.001------
accuracy: 0.7659963436928702
[[187  87]
 [ 41 232]]
balanced accuracy: 0.7661493008208337
```

```
-----0.0001------
accuracy: 0.7678244972577697
[[167 107]
 [ 20 253]]
balanced accuracy: 0.7681144889174086
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************46*****************
-----0.01------
accuracy: 0.7458866544789763
[[179  95]
 [ 44 229]]
balanced accuracy: 0.7460562551803428
-----0.001------
accuracy: 0.7495429616087751
[[174 100]
 [ 37 236]]
balanced accuracy: 0.7497526804096147
-----0.0001------
accuracy: 0.7568555758683729
[[164 110]
 [ 23 250]]
balanced accuracy: 0.7571455308681586
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************47*****************
-----0.01------
accuracy: 0.7769652650822669
[[188  86]
 [ 36 237]]
balanced accuracy: 0.777131627496591
-----0.001------
accuracy: 0.7659963436928702
[[180  94]
 [ 34 239]]
balanced accuracy: 0.7661960910136092
-----0.0001------
accuracy: 0.7787934186471663
[[175  99]
 [ 22 251]]
balanced accuracy: 0.7790500254003904
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************48****************
-----0.01------
accuracy: 0.7879341864716636
[[201  73]
 [ 43 230]]
balanced accuracy: 0.7880337424133045
-----0.001------
accuracy: 0.7824497257769653
[[198  76]
 [ 43 230]]
balanced accuracy: 0.7825592898585599
-----0.0001------
accuracy: 0.7824497257769653
[[170 104]
 [ 15 258]]
balanced accuracy: 0.7827464506296623
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************49****************
-----0.01------
accuracy: 0.7897623400365631
[[200  74]
 [ 41 232]]
balanced accuracy: 0.78987192855806
-----0.001------
accuracy: 0.8007312614259597
[[198  76]
 [ 33 240]]
balanced accuracy: 0.8008743081735783
-----0.0001------
accuracy: 0.7970749542961609
[[183  91]
 [ 20 253]]
balanced accuracy: 0.7973115692093795
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************50****************
-----0.01------
```

```
accuracy: 0.7513711151736746
[[190  84]
 [ 52 221]]
balanced accuracy: 0.751477233229058
-----0.001------
accuracy: 0.7605118829981719
[[188  86]
 [ 45 228]]
balanced accuracy: 0.7606481110130745
-----0.0001------
accuracy: 0.773308957952468
[[176  98]
 [ 26 247]]
balanced accuracy: 0.7735488355926312
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************51*****************
-----0.01------
accuracy: 0.7769652650822669
[[188  86]
 [ 36 237]]
balanced accuracy: 0.777131627496591
-----0.001------
accuracy: 0.7641681901279708
[[184  90]
 [ 39 234]]
balanced accuracy: 0.7643378519290928
-----0.0001------
accuracy: 0.7751371115173674
[[170 104]
 [ 19 254]]
balanced accuracy: 0.775420443303655
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************52*****************
-----0.01------
accuracy: 0.7678244972577697
[[188  86]
 [ 41 232]]
balanced accuracy: 0.7679741183390818
-----0.001------
accuracy: 0.7678244972577697
```

```
[[181  93]
 [ 34 239]]
balanced accuracy: 0.7680209085318574
-----0.0001------
accuracy: 0.7678244972577697
[[169 105]
 [ 22 251]]
balanced accuracy: 0.7681011202909013
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************53****************
-----0.01------
accuracy: 0.7806215722120659
[[188  86]
 [ 34 239]]
balanced accuracy: 0.7807946311595947
-----0.001------
accuracy: 0.7550274223034735
[[172 102]
 [ 32 241]]
balanced accuracy: 0.7552605545306275
-----0.0001------
accuracy: 0.7659963436928702
[[170 104]
 [ 24 249]]
balanced accuracy: 0.7662629341461458
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************54****************
-----0.01------
accuracy: 0.7605118829981719
[[201  73]
 [ 58 215]]
balanced accuracy: 0.760561214940777
-----0.001------
accuracy: 0.7641681901279708
[[185  89]
 [ 40 233]]
balanced accuracy: 0.7643311676158391
-----0.0001------
accuracy: 0.7678244972577697
[[180  94]
```

```
 [ 33 240]]
balanced accuracy: 0.768027592845111
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************55****************
-----0.01------
accuracy: 0.7641681901279708
[[202  72]
 [ 57 216]]
balanced accuracy: 0.764217534290527
-----0.001------
accuracy: 0.7623400365630713
[[188  86]
 [ 44 229]]
balanced accuracy: 0.7624796128445763
-----0.0001------
accuracy: 0.7714808043875686
[[171 103]
 [ 22 251]]
balanced accuracy: 0.7717507553273977
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************56****************
-----0.01------
accuracy: 0.753199268738574
[[179  95]
 [ 40 233]]
balanced accuracy: 0.7533822625063501
-----0.001------
accuracy: 0.7513711151736746
[[166 108]
 [ 28 245]]
balanced accuracy: 0.7516376567471459
-----0.0001------
accuracy: 0.7513711151736746
[[156 118]
 [ 18 255]]
balanced accuracy: 0.7517044998796824
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
*************57****************
-----0.01------
accuracy: 0.7659963436928702
[[184  90]
 [ 38 235]]
balanced accuracy: 0.7661693537605947
-----0.001------
accuracy: 0.753199268738574
[[188  86]
 [ 49 224]]
balanced accuracy: 0.7533221036870672
-----0.0001------
accuracy: 0.7696526508226691
[[169 105]
 [ 21 252]]
balanced accuracy: 0.7699326221224032
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************58****************
-----0.01------
accuracy: 0.7915904936014625
[[198  76]
 [ 38 235]]
balanced accuracy: 0.7917167990160692
-----0.001------
accuracy: 0.7970749542961609
[[198  76]
 [ 35 238]]
balanced accuracy: 0.7972113045105746
-----0.0001------
accuracy: 0.7989031078610603
[[193  81]
 [ 29 244]]
balanced accuracy: 0.7990762279083448
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************59****************
-----0.01------
accuracy: 0.7751371115173674
[[201  73]
 [ 50 223]]
```

```
balanced accuracy: 0.7752132295927916
-----0.001------
accuracy: 0.7806215722120659
[[195  79]
 [ 41 232]]
balanced accuracy: 0.780747840966819
-----0.0001------
accuracy: 0.7769652650822669
[[179  95]
 [ 27 246]]
balanced accuracy: 0.7771917863158739
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************60*****************
-----0.01------
accuracy: 0.7897623400365631
[[194  80]
 [ 35 238]]
balanced accuracy: 0.7899120344375818
-----0.001------
accuracy: 0.793418647166362
[[192  82]
 [ 31 242]]
balanced accuracy: 0.7935884067270929
-----0.0001------
accuracy: 0.793418647166362
[[184  90]
 [ 23 250]]
balanced accuracy: 0.7936418812331221
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************61*****************
-----0.01------
accuracy: 0.7751371115173674
[[187  87]
 [ 36 237]]
balanced accuracy: 0.7753068099783429
-----0.001------
accuracy: 0.7769652650822669
[[188  86]
 [ 36 237]]
balanced accuracy: 0.777131627496591
```

```
-----0.0001------
accuracy: 0.7842778793418648
[[179  95]
 [ 23 250]]
balanced accuracy: 0.7845177936418812
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************62*****************
-----0.01------
accuracy: 0.7915904936014625
[[199  75]
 [ 39 234]]
balanced accuracy: 0.7917101147028154
-----0.001------
accuracy: 0.7952468007312614
[[198  76]
 [ 36 237]]
balanced accuracy: 0.7953798026790728
-----0.0001------
accuracy: 0.7915904936014625
[[182  92]
 [ 22 251]]
balanced accuracy: 0.7918237480281276
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************63*****************
-----0.01------
accuracy: 0.7129798903107861
[[175  99]
 [ 58 215]]
balanced accuracy: 0.7131159594663244
-----0.001------
accuracy: 0.7221206581352834
[[168 106]
 [ 46 227]]
balanced accuracy: 0.7223202588166092
-----0.0001------
accuracy: 0.7257769652650823
[[144 130]
 [ 20 253]]
balanced accuracy: 0.7261436859977006
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************64****************
-----0.01------
accuracy: 0.7897623400365631
[[202  72]
 [ 43 230]]
balanced accuracy: 0.7898585599315526
-----0.001------
accuracy: 0.7897623400365631
[[198  76]
 [ 39 234]]
balanced accuracy: 0.7898852971845672
-----0.0001------
accuracy: 0.7952468007312614
[[187  87]
 [ 25 248]]
balanced accuracy: 0.795453330124863
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************65****************
-----0.01------
accuracy: 0.7678244972577697
[[183  91]
 [ 36 237]]
balanced accuracy: 0.7680075399053501
-----0.001------
accuracy: 0.7513711151736746
[[181  93]
 [ 43 230]]
balanced accuracy: 0.751537392048341
-----0.0001------
accuracy: 0.7769652650822669
[[163 111]
 [ 11 262]]
balanced accuracy: 0.7772987353279324
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************66****************
-----0.01------
```

```
accuracy: 0.7605118829981719
[[202  72]
 [ 59 214]]
balanced accuracy: 0.7605545306275233
-----0.001------
accuracy: 0.7678244972577697
[[190  84]
 [ 43 230]]
balanced accuracy: 0.7679607497125746
-----0.0001------
accuracy: 0.7915904936014625
[[180  94]
 [ 20 253]]
balanced accuracy: 0.791837116654635
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************67*****************
-----0.01------
accuracy: 0.7586837294332724
[[194  80]
 [ 52 221]]
balanced accuracy: 0.7587765033020508
-----0.001------
accuracy: 0.7696526508226691
[[180  94]
 [ 32 241]]
balanced accuracy: 0.7698590946766128
-----0.0001------
accuracy: 0.7769652650822669
[[171 103]
 [ 19 254]]
balanced accuracy: 0.7772452608219032
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************68*****************
-----0.01------
accuracy: 0.7787934186471663
[[191  83]
 [ 38 235]]
balanced accuracy: 0.7789430763883318
-----0.001------
accuracy: 0.7897623400365631
```

```
[[190  84]
 [ 31 242]]
balanced accuracy: 0.7899387716905966
-----0.0001------
accuracy: 0.7861060329067642
[[176  98]
 [ 19 254]]
balanced accuracy: 0.786369348413144
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************69****************
-----0.01------
accuracy: 0.773308957952468
[[188  86]
 [ 38 235]]
balanced accuracy: 0.7734686238335874
-----0.001------
accuracy: 0.7641681901279708
[[173 101]
 [ 28 245]]
balanced accuracy: 0.764411379374883
-----0.0001------
accuracy: 0.7787934186471663
[[177  97]
 [ 24 249]]
balanced accuracy: 0.7790366567738831
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************70****************
-----0.01------
accuracy: 0.7879341864716636
[[199  75]
 [ 41 232]]
balanced accuracy: 0.7880471110398117
-----0.001------
accuracy: 0.7678244972577697
[[191  83]
 [ 44 229]]
balanced accuracy: 0.7679540653993209
-----0.0001------
accuracy: 0.7970749542961609
[[187  87]
```

```
 [ 24 249]]
balanced accuracy: 0.7972848319563648
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************71*****************
-----0.01------
accuracy: 0.7586837294332724
[[180  94]
 [ 38 235]]
balanced accuracy: 0.7588700836876019
-----0.001------
accuracy: 0.753199268738574
[[175  99]
 [ 36 237]]
balanced accuracy: 0.7534089997593647
-----0.0001------
accuracy: 0.7477148080438757
[[147 127]
 [ 11 262]]
balanced accuracy: 0.7481016550359616
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************72*****************
-----0.01------
accuracy: 0.7787934186471663
[[185  89]
 [ 32 241]]
balanced accuracy: 0.7789831822678538
-----0.001------
accuracy: 0.7861060329067642
[[185  89]
 [ 28 245]]
balanced accuracy: 0.7863091895938612
-----0.0001------
accuracy: 0.7751371115173674
[[175  99]
 [ 24 249]]
balanced accuracy: 0.7753870217373867
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
*************73****************
-----0.01------
accuracy: 0.7787934186471663
[[198  76]
 [ 45 228]]
balanced accuracy: 0.7788962861955563
-----0.001------
accuracy: 0.7824497257769653
[[193  81]
 [ 38 235]]
balanced accuracy: 0.7825927114248282
-----0.0001------
accuracy: 0.7806215722120659
[[182  92]
 [ 28 245]]
balanced accuracy: 0.7808347370391167
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************74****************
-----0.01------
accuracy: 0.7659963436928702
[[197  77]
 [ 51 222]]
balanced accuracy: 0.7660824576882971
-----0.001------
accuracy: 0.7659963436928702
[[182  92]
 [ 36 237]]
balanced accuracy: 0.7661827223871019
-----0.0001------
accuracy: 0.7641681901279708
[[168 106]
 [ 23 250]]
balanced accuracy: 0.7644448009411513
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************75****************
-----0.01------
accuracy: 0.7495429616087751
[[187  87]
 [ 50 223]]
```

```
balanced accuracy: 0.7496657843373171
-----0.001------
accuracy: 0.7586837294332724
[[179  95]
 [ 37 236]]
balanced accuracy: 0.7588767680008557
-----0.0001------
accuracy: 0.7678244972577697
[[164 110]
 [ 17 256]]
balanced accuracy: 0.7681345418571697
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************76*****************
-----0.01------
accuracy: 0.773308957952468
[[187  87]
 [ 37 236]]
balanced accuracy: 0.7734753081468411
-----0.001------
accuracy: 0.7751371115173674
[[185  89]
 [ 34 239]]
balanced accuracy: 0.7753201786048501
-----0.0001------
accuracy: 0.7678244972577697
[[164 110]
 [ 17 256]]
balanced accuracy: 0.7681345418571697
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************77*****************
-----0.01------
accuracy: 0.7659963436928702
[[189  85]
 [ 43 230]]
balanced accuracy: 0.7661359321943264
-----0.001------
accuracy: 0.7678244972577697
[[179  95]
 [ 32 241]]
balanced accuracy: 0.7680342771583648
```

```
-----0.0001------
accuracy: 0.7477148080438757
[[158 116]
 [ 22 251]]
balanced accuracy: 0.7480281275901713
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************78****************
-----0.01------
accuracy: 0.7714808043875686
[[191  83]
 [ 42 231]]
balanced accuracy: 0.7716170690623245
-----0.001------
accuracy: 0.753199268738574
[[178  96]
 [ 39 234]]
balanced accuracy: 0.7533889468196038
-----0.0001------
accuracy: 0.773308957952468
[[177  97]
 [ 27 246]]
balanced accuracy: 0.7735421512793776
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************79****************
-----0.01------
accuracy: 0.7586837294332724
[[195  79]
 [ 53 220]]
balanced accuracy: 0.758769818988797
-----0.001------
accuracy: 0.7714808043875686
[[186  88]
 [ 37 236]]
balanced accuracy: 0.7716504906285928
-----0.0001------
accuracy: 0.7769652650822669
[[175  99]
 [ 23 250]]
balanced accuracy: 0.7772185235688885
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************80****************
-----0.01------
accuracy: 0.7696526508226691
[[191  83]
 [ 43 230]]
balanced accuracy: 0.7697855672308227
-----0.001------
accuracy: 0.7641681901279708
[[185  89]
 [ 40 233]]
balanced accuracy: 0.7643311676158391
-----0.0001------
accuracy: 0.7751371115173674
[[170 104]
 [ 19 254]]
balanced accuracy: 0.775420443303655
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************81****************
-----0.01------
accuracy: 0.7824497257769653
[[188  86]
 [ 33 240]]
balanced accuracy: 0.7826261329910965
-----0.001------
accuracy: 0.773308957952468
[[189  85]
 [ 39 234]]
balanced accuracy: 0.7734619395203337
-----0.0001------
accuracy: 0.7659963436928702
[[163 111]
 [ 17 256]]
balanced accuracy: 0.7663097243389214
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************82****************
-----0.01------
```

```
accuracy: 0.7824497257769653
[[197  77]
 [ 42 231]]
balanced accuracy: 0.7825659741718136
-----0.001------
accuracy: 0.7861060329067642
[[184  90]
 [ 27 246]]
balanced accuracy: 0.7863158739071148
-----0.0001------
accuracy: 0.7751371115173674
[[170 104]
 [ 19 254]]
balanced accuracy: 0.775420443303655
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************83****************
-----0.01------
accuracy: 0.7623400365630713
[[198  76]
 [ 54 219]]
balanced accuracy: 0.7624127697120398
-----0.001------
accuracy: 0.7842778793418648
[[197  77]
 [ 41 232]]
balanced accuracy: 0.7843974760033154
-----0.0001------
accuracy: 0.7824497257769653
[[174 100]
 [ 19 254]]
balanced accuracy: 0.7827197133766477
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************84****************
-----0.01------
accuracy: 0.773308957952468
[[192  82]
 [ 42 231]]
balanced accuracy: 0.7734418865805728
-----0.001------
accuracy: 0.773308957952468
```

```
[[180  94]
 [ 30 243]]
balanced accuracy: 0.7735220983396165
-----0.0001------
accuracy: 0.7751371115173674
[[165 109]
 [ 14 259]]
balanced accuracy: 0.7754538648699232
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************85****************
-----0.01------
accuracy: 0.7659963436928702
[[185  89]
 [ 39 234]]
balanced accuracy: 0.7661626694473409
-----0.001------
accuracy: 0.7623400365630713
[[187  87]
 [ 43 230]]
balanced accuracy: 0.76248629715783
-----0.0001------
accuracy: 0.773308957952468
[[169 105]
 [ 19 254]]
balanced accuracy: 0.7735956257854069
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************86****************
-----0.01------
accuracy: 0.7824497257769653
[[187  87]
 [ 32 241]]
balanced accuracy: 0.7826328173043502
-----0.001------
accuracy: 0.7641681901279708
[[182  92]
 [ 37 236]]
balanced accuracy: 0.7643512205556001
-----0.0001------
accuracy: 0.7751371115173674
[[170 104]
```

```
 [ 19 254]]
balanced accuracy: 0.775420443303655
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************87****************
-----0.01------
accuracy: 0.7641681901279708
[[185  89]
 [ 40 233]]
balanced accuracy: 0.7643311676158391
-----0.001------
accuracy: 0.7477148080438757
[[168 106]
 [ 32 241]]
balanced accuracy: 0.7479612844576349
-----0.0001------
accuracy: 0.773308957952468
[[169 105]
 [ 19 254]]
balanced accuracy: 0.7735956257854069
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************88****************
-----0.01------
accuracy: 0.7842778793418648
[[196  78]
 [ 40 233]]
balanced accuracy: 0.784404160316569
-----0.001------
accuracy: 0.7842778793418648
[[202  72]
 [ 46 227]]
balanced accuracy: 0.7843640544370472
-----0.0001------
accuracy: 0.7861060329067642
[[177  97]
 [ 20 253]]
balanced accuracy: 0.7863626640998904
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
```

```
balanced accuracy: 0.5
*************89****************
-----0.01------
accuracy: 0.7495429616087751
[[181  93]
 [ 44 229]]
balanced accuracy: 0.7497058902168392
-----0.001------
accuracy: 0.753199268738574
[[180  94]
 [ 41 232]]
balanced accuracy: 0.7533755781930964
-----0.0001------
accuracy: 0.7495429616087751
[[165 109]
 [ 28 245]]
balanced accuracy: 0.7498128392288976
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************90****************
-----0.01------
accuracy: 0.7824497257769653
[[191  83]
 [ 36 237]]
balanced accuracy: 0.7826060800513355
-----0.001------
accuracy: 0.7842778793418648
[[191  83]
 [ 35 238]]
balanced accuracy: 0.7844375818828373
-----0.0001------
accuracy: 0.7769652650822669
[[170 104]
 [ 18 255]]
balanced accuracy: 0.7772519451351568
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************91****************
-----0.01------
accuracy: 0.7659963436928702
[[185  89]
 [ 39 234]]
```

```
balanced accuracy: 0.7661626694473409
-----0.001------
accuracy: 0.7605118829981719
[[187  87]
 [ 44 229]]
balanced accuracy: 0.7606547953263282
-----0.0001------
accuracy: 0.7897623400365631
[[177  97]
 [ 18 255]]
balanced accuracy: 0.7900256677628941
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************92*****************
-----0.01------
accuracy: 0.8190127970749543
[[217  57]
 [ 42 231]]
balanced accuracy: 0.8190623245367771
-----0.001------
accuracy: 0.8080438756855576
[[206  68]
 [ 37 236]]
balanced accuracy: 0.8081468409935564
-----0.0001------
accuracy: 0.8171846435100548
[[196  78]
 [ 22 251]]
balanced accuracy: 0.8173711932836021
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
**************93*****************
-----0.01------
accuracy: 0.7696526508226691
[[189  85]
 [ 41 232]]
balanced accuracy: 0.7697989358573301
-----0.001------
accuracy: 0.7714808043875686
[[191  83]
 [ 42 231]]
balanced accuracy: 0.7716170690623245
```

```
-----0.0001------
accuracy: 0.7696526508226691
[[168 106]
 [ 20 253]]
balanced accuracy: 0.7699393064356568
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************94****************
-----0.01------
accuracy: 0.7897623400365631
[[202  72]
 [ 43 230]]
balanced accuracy: 0.7898585599315526
-----0.001------
accuracy: 0.7861060329067642
[[195  79]
 [ 38 235]]
balanced accuracy: 0.7862423464613246
-----0.0001------
accuracy: 0.7879341864716636
[[175  99]
 [ 17 256]]
balanced accuracy: 0.7882075345578996
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************95****************
-----0.01------
accuracy: 0.7659963436928702
[[186  88]
 [ 40 233]]
balanced accuracy: 0.7661559851340873
-----0.001------
accuracy: 0.7678244972577697
[[186  88]
 [ 39 234]]
balanced accuracy: 0.7679874869655892
-----0.0001------
accuracy: 0.7714808043875686
[[179  95]
 [ 30 243]]
balanced accuracy: 0.7716972808213685
-----1e-05------
```

```
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************96****************
-----0.01------
accuracy: 0.7659963436928702
[[188  86]
 [ 42 231]]
balanced accuracy: 0.76614261650758
-----0.001------
accuracy: 0.7806215722120659
[[193  81]
 [ 39 234]]
balanced accuracy: 0.7807612095933263
-----0.0001------
accuracy: 0.7806215722120659
[[172 102]
 [ 18 255]]
balanced accuracy: 0.7809015801716532
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************97****************
-----0.01------
accuracy: 0.7824497257769653
[[198  76]
 [ 43 230]]
balanced accuracy: 0.7825592898585599
-----0.001------
accuracy: 0.7824497257769653
[[197  77]
 [ 42 231]]
balanced accuracy: 0.7825659741718136
-----0.0001------
accuracy: 0.773308957952468
[[178  96]
 [ 28 245]]
balanced accuracy: 0.7735354669661239
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************98****************
-----0.01------
```

```
accuracy: 0.7915904936014625
[[196  78]
 [ 36 237]]
balanced accuracy: 0.7917301676425764
-----0.001------
accuracy: 0.7861060329067642
[[196  78]
 [ 39 234]]
balanced accuracy: 0.7862356621480708
-----0.0001------
accuracy: 0.7915904936014625
[[178  96]
 [ 18 255]]
balanced accuracy: 0.7918504852811422
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
*************99****************
-----0.01------
accuracy: 0.7824497257769653
[[192  82]
 [ 37 236]]
balanced accuracy: 0.7825993957380819
-----0.001------
accuracy: 0.7769652650822669
[[189  85]
 [ 37 236]]
balanced accuracy: 0.7771249431833374
-----0.0001------
accuracy: 0.7659963436928702
[[166 108]
 [ 20 253]]
balanced accuracy: 0.7662896713991605
-----1e-05------
accuracy: 0.4990859232175503
[[  0 274]
 [  0 273]]
balanced accuracy: 0.5
```

3.2 ensemble

3.2.1 voting

[ ]:
```
# ensemble
x=data2.drop(columns=['target'])
y=data2['target']*1
ratio=y.value_counts()[0]/y.value_counts()[1]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42, stratify=y)
# x_test=x
# y_test=y

# voting
preds=[]
for model in models:
    # predict on test data
    pred_test = model.predict(x_test)
    preds.append(pred_test)
preds=np.array(preds)
preds=preds.sum(axis=0) # n_samples, range(0,100)
thresh=90
preds[preds<thresh]=0
preds[preds>=thresh]=1

# compute accuracy
print(f"voting accuracy: {accuracy_score(y_test, preds)}")

# confusion matrix
print(confusion_matrix(y_test, preds))

# compute balanced accuracy
acc=balanced_accuracy_score(y_test, preds)
print(
    f"voting balanced accuracy: {acc}"
)
```

```
voting accuracy: 0.7718675168832566
[[28459  8435]
 [   44   229]]
voting balanced accuracy: 0.805099938820869
```

3.2.2 linear combination

[ ]:
```
# ensemble
from sklearn.linear_model import LogisticRegression
```

```
x=data2.drop(columns=['target'])
y=data2['target']*1
ratio=y.value_counts()[0]/y.value_counts()[1]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)

# linera combination
preds=[]
for model in models:
    # predict on test data
    pred_train = model.predict(x_train)
    preds.append(pred_train)
preds=np.array(preds).T # n_samples*100

combine_model=LogisticRegression(random_state=42,class_weight={0: sum(y_train)/
 ↪len(y_train),1: 1-sum(y_train)/len(y_train)}).fit(preds, y_train)

preds=[]
for model in models:
    # predict on test data
    pred_test = model.predict(x_test)
    preds.append(pred_test)
preds=np.array(preds).T # n_samples*100

# compute accuracy
pred_test = combine_model.predict(preds)
print(f"linear accuracy: {accuracy_score(y_test, pred_test)}")

# confusion matrix
print(confusion_matrix(y_test, pred_test))

# compute balanced accuracy
acc=balanced_accuracy_score(y_test, pred_test)
print(
    f"linear balanced accuracy: {acc}"
)
```

```
linear accuracy: 0.7510156859579735
[[27673  9221]
 [   33   240]]
linear balanced accuracy: 0.814594320408274
```

4 Interpretation

[ ]:
```
import shap

# 100 models overall importance
# top 20? important ensembled models, feture importance
# top 20? models, 4 categories (TP, TN, FP, FN)
# top 20 models, tree visualization
```

4.1 100 models overall importance

[ ]:
```
def getFeatureImportance(model, cur_data):
  shap_values = shap.TreeExplainer(model.booster_).shap_values(cur_data)
  importance=np.mean(shap_values[0], axis=0)
  importance=importance/np.sum(importance)
  name=list(cur_data.columns)
  return name, importance
```

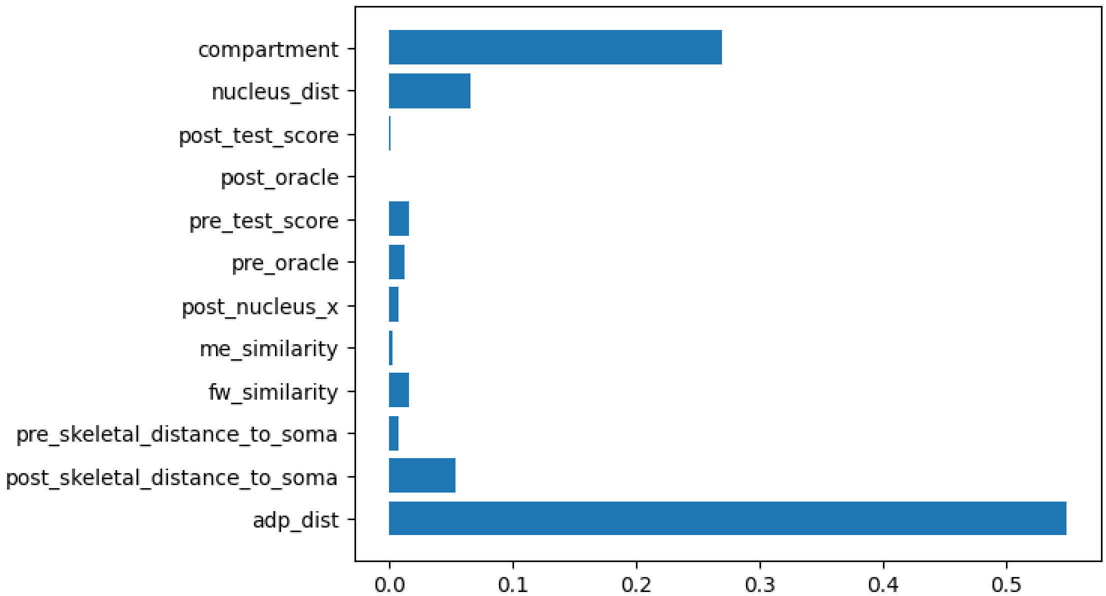
[ ]:
```
feature_names=[]
importances=[]
shap.initjs()
for model in models:
  feature_names, importance=getFeatureImportance(model, x_test)
  importances.append(importance)
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
importances2=np.array(importances)
```

[ ]:
```
importances2=np.mean(importances2, axis=0)
```

[ ]:
```
bars =feature_names
y_pos = np.arange(len(importances2))

# Create horizontal bars
plt.barh(y_pos, importances2)

# Create names on the x-axis
plt.yticks(y_pos, bars)

# Show graphic
plt.show()
```



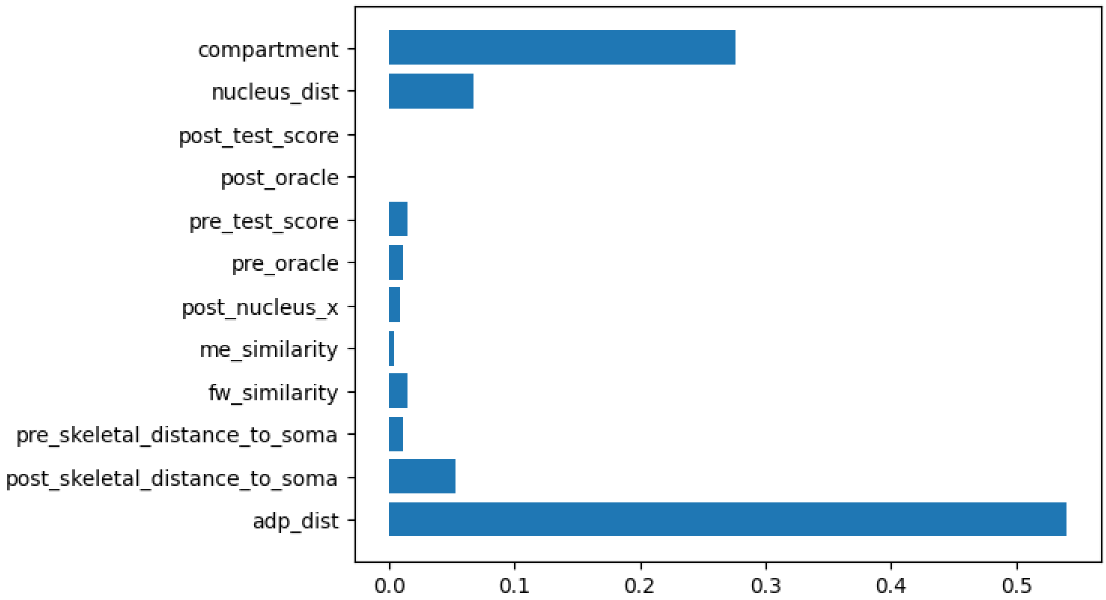
4.2 top 10 models

[ ]:
```
model_importance=abs(combine_model.coef_[0])
```

[ ]:
```
model_indices=np.argsort(-model_importance)
```

[ ]:
```
importances2=np.array(importances)[model_indices[:10]]
importances2=np.mean(importances2, axis=0)
```

[ ]:
```
bars =feature_names
y_pos = np.arange(len(importances2))

# Create horizontal bars
plt.barh(y_pos, importances2)

# Create names on the x-axis
plt.yticks(y_pos, bars)

# Show graphic
plt.show()
```



4.3 True prediction VS False prediction

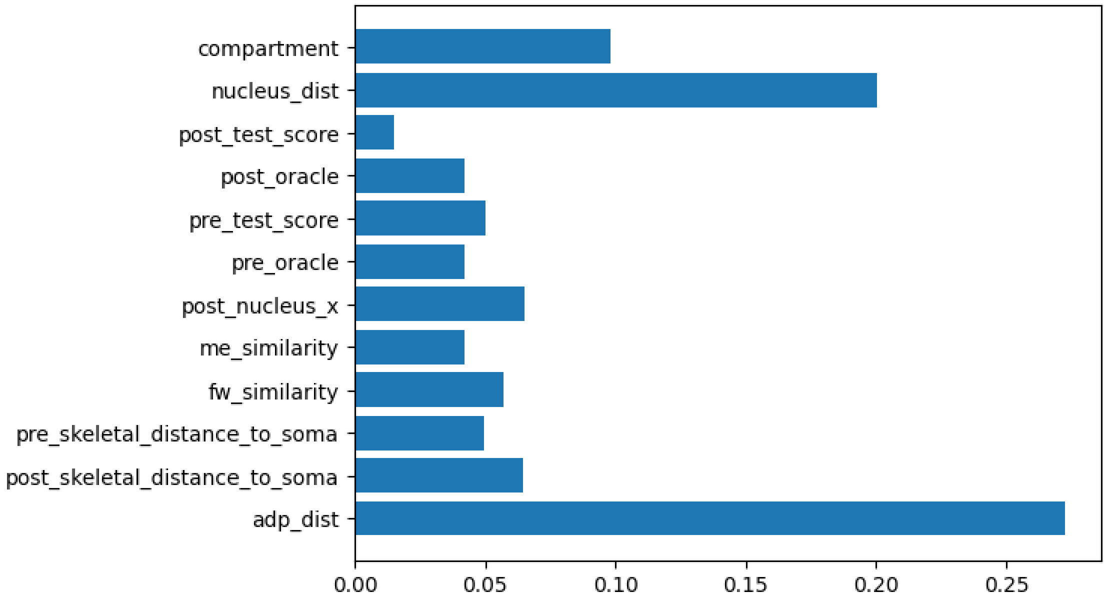
[ ]:
```
false_prediction_indices=np.where(pred_test!=y_test)[0]
true_prediciton_indices=np.where(pred_test==y_test)[0]

positive_indices=np.where(y_test==1)[0]
neg_indices=np.where(y_test==0)[0]

tp=np.intersect1d(true_prediciton_indices, positive_indices)
tn=np.intersect1d(true_prediciton_indices, neg_indices)
fp=np.intersect1d(false_prediction_indices, positive_indices)
fn=np.intersect1d(false_prediction_indices, neg_indices)

print("tp:", len(tp))
print("tn:", len(tn))
print("fp:", len(fp))
print("fn:", len(fn))
```

```
tp: 240
tn: 27673
fp: 33
fn: 9221
```

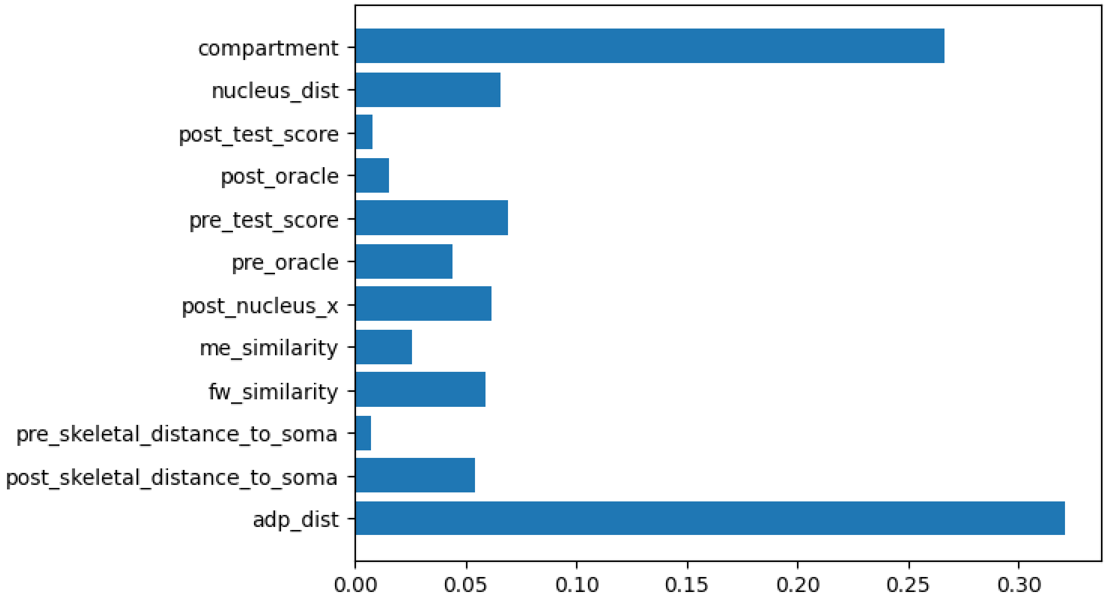
[ ]:
```
np.random.seed(42)
cate=["tp", "tn", "fp", "fn"]
for cur_data, cur_cate in zip([tp, tn, fp, fn], cate):
  print("********************")
```

```
print(cur_cate)
print("********************")
cur_cate_importances=[]
for cur_index in np.random.choice(cur_data, 10, replace=False):
  name, importance = getFeatureImportance(models[model_indices[0]], x_test.
↪iloc[[cur_index]])
  importance=abs(importance)/np.sum(abs(importance))
  cur_cate_importances.append(importance)
cur_cate_importances=np.array(cur_cate_importances)
cur_cate_importances=np.mean(cur_cate_importances, axis=0)

bars =name
y_pos = np.arange(len(cur_cate_importances))

# Create horizontal bars
plt.barh(y_pos, cur_cate_importances)

# Create names on the x-axis
plt.yticks(y_pos, bars)

# Show graphic
plt.show()
```

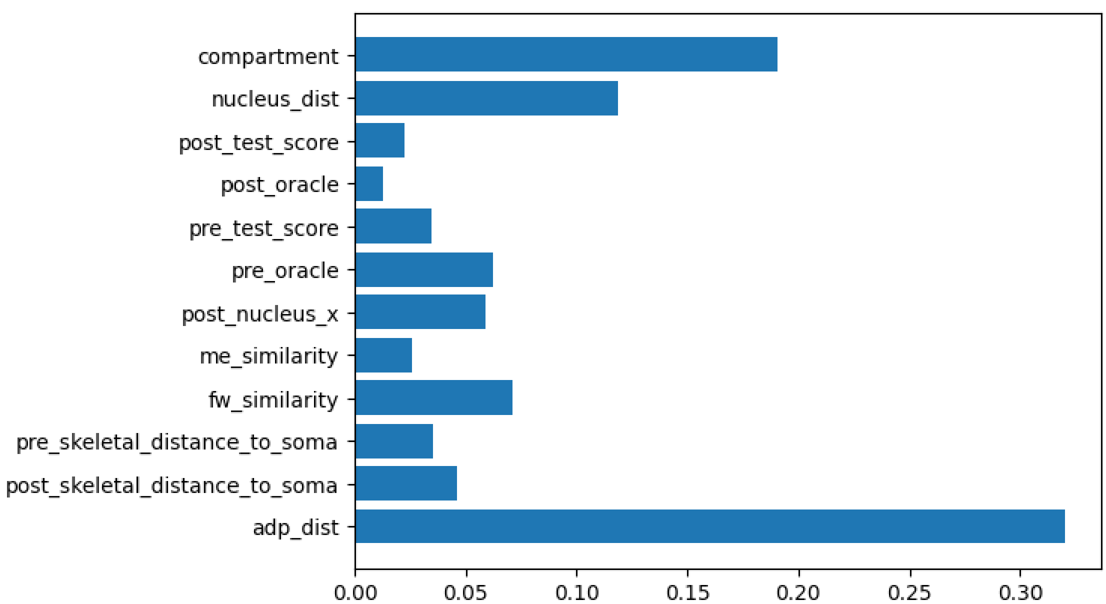
```
********************
tp
********************
```



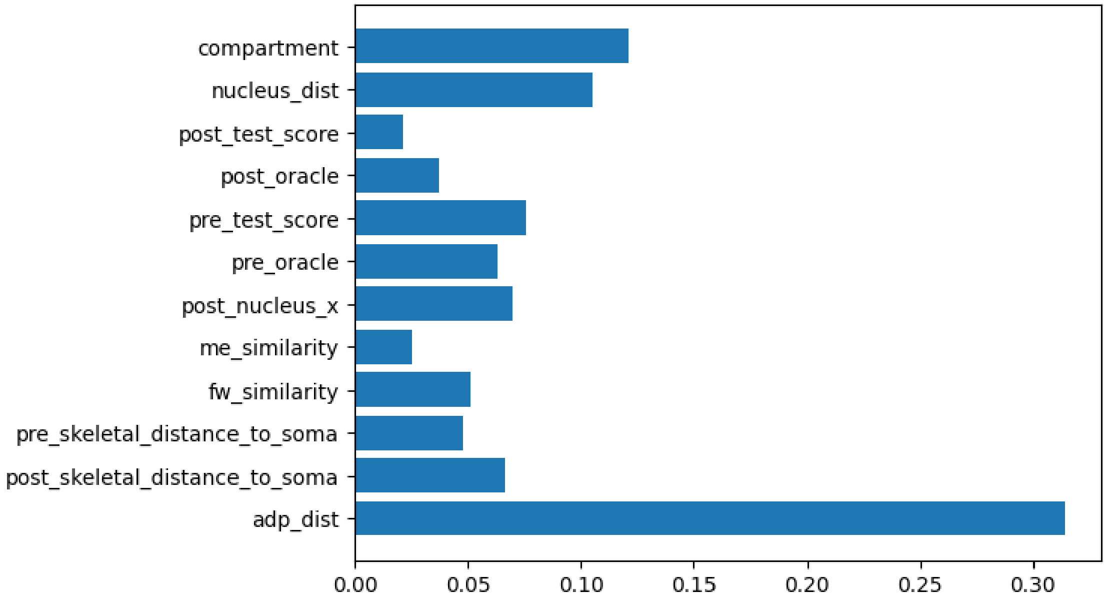
****
tn
****


compartment
nucleus_dist
post_test_score
post_oracle
pre_test_score
pre_oracle
post_nucleus_x
me_similarity
fw_similarity
pre_skeletal_distance_to_soma
post_skeletal_distance_to_soma
adp_dist
0.00
0.05
0.10
0.15
0.20
0.25
0.30

****
fp
****


compartment
nucleus_dist
post_test_score
post_oracle
pre_test_score
pre_oracle
post_nucleus_x
me_similarity
fw_similarity
pre_skeletal_distance_to_soma
post_skeletal_distance_to_soma
adp_dist
0.00
0.05
0.10
0.15
0.20
0.25
0.30

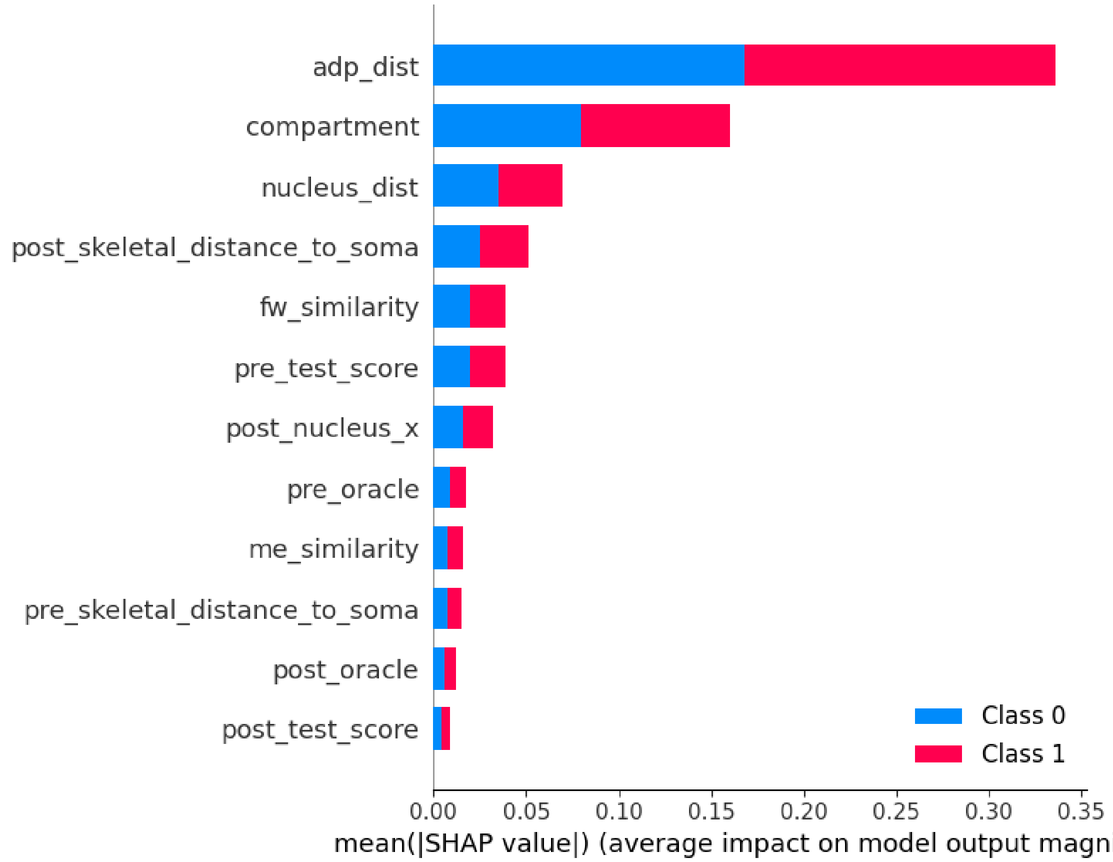
```
********************
fn
********************
```



4.4 Shap Example

[ ]:
```
shap.initjs()
shap_values = shap.TreeExplainer(models[0].booster_).shap_values(x)
shap.summary_plot(shap_values, x, show=True)
# plt.savefig('feature importance.pdf', bbox_inches='tight')
```

```
<IPython.core.display.HTML object>
```

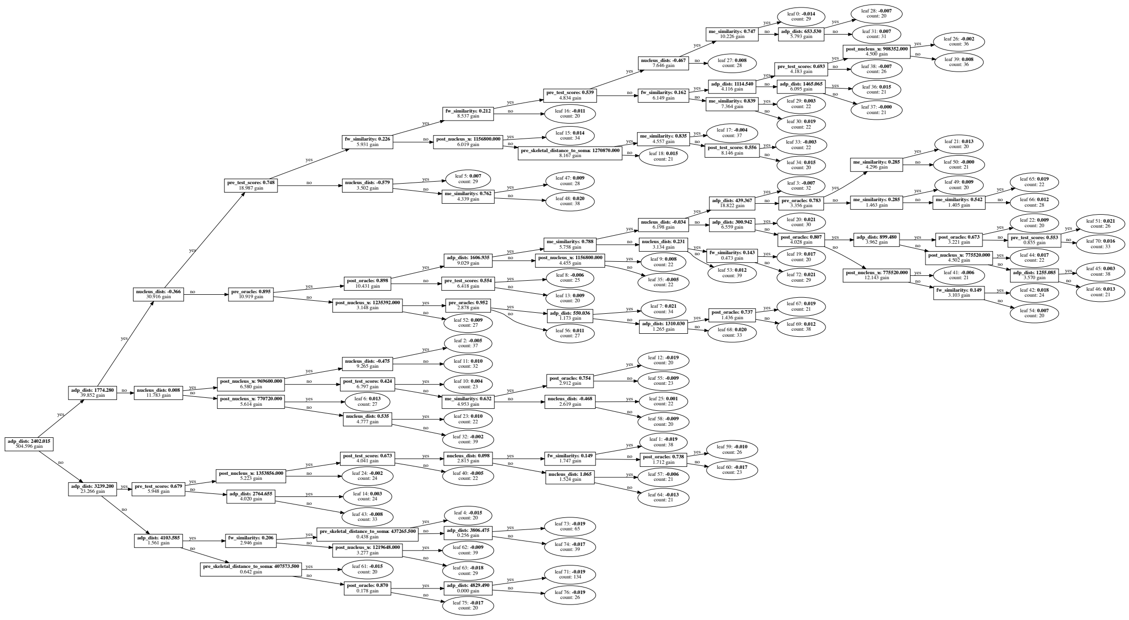


[ ]:
```
!pwd
```

```
/content/drive/Shareddrives/578_term
```

[ ]:
```
# display and save decision tree
dot=lgb.create_tree_digraph(models[1], tree_index=0,␣
↪show_info=['split_gain','leaf_count']) # get display figure
dot.render(directory='treeVisualization2', view=True)   # save
dot # show
```

[ ]:



5 predict

5.0.1 Load and Merge Leaderboard Data

```
[ ]: lb_data = pd.read_csv("./data/leaderboard_data.csv")
lb_data = (
    lb_data.merge(
        feature_weights.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        feature_weights.rename(columns=lambda x: "post_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "pre_" + x),
        how="left",
        validate="m:1",
        copy=False,
    )
    .merge(
        morph_embeddings.rename(columns=lambda x: "post_" + x),
```

```
        how="left",
        validate="m:1",
        copy=False,
    )
)
```

[ ]:
```
# compute the cosine similarity between the pre- and post- feature weights
lb_data["fw_similarity"] = lb_data.apply(row_feature_similarity, axis=1)
lb_data["me_similarity"] = lb_data.apply(row_morph_similarity, axis=1)
```

[ ]:
```
drop_columns=['pre_feature_weights', 'post_feature_weights',␣
↪'pre_morph_embeddings', 'post_morph_embeddings', 'pre_nucleus_id',␣
↪'post_nucleus_id', 'ID' ]

remain_columns=['adp_dist','post_skeletal_distance_to_soma',␣
↪'pre_skeletal_distance_to_soma','fw_similarity', 'me_similarity',␣
↪'post_nucleus_x']

relation_onehot_columns=[
    ['pre_brain_area', 'post_brain_area'], # 1. use xor to get new feature.  2.␣
↪no, found out connected neurons can located in both.
                        ]
relation_number_columns=[
    ['pre_nucleus_x','pre_nucleus_y','pre_nucleus_z', 'post_nucleus_y',␣
↪'post_nucleus_z'] # get the distance between pre and post
]
question_columns=['pre_oracle', 'pre_test_score', 'post_oracle',␣
↪'post_test_score'] # oracle relates to nueron reliability, test_score␣
↪relates to similarity.

target_columns=['connected']
```

[ ]:
```
remain_columns.extend(question_columns)
lb_data2=lb_data[remain_columns].copy(deep=True)
```

[ ]:
```
lb_data2["nucleus_dist"]=abs(lb_data['pre_nucleus_x']-lb_data['post_nucleus_x'])**2+abs(lb_dat
lb_data2["nucleus_dist"]=(lb_data2["nucleus_dist"]-lb_data2["nucleus_dist"].
↪mean())/lb_data2["nucleus_dist"].std()

# lb_data2["rf_dist"] =␣
↪abs(lb_data['axonal_coor_x']-lb_data['post_rf_x'])**2+abs(lb_data['axonal_coor_y']-lb_data[
# lb_data2["rf_dist"]=(lb_data2["rf_dist"]-lb_data2["rf_dist"].mean())/
↪lb_data2["rf_dist"].std()

lb_data2['compartment']=lb_data['compartment'].astype("category")

lb_data2['ID']=lb_data['ID']
```

```
lb_data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 42593 entries, 0 to 42592
Data columns (total 13 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   adp_dist                       42593 non-null  float64
 1   post_skeletal_distance_to_soma 42593 non-null  float64
 2   pre_skeletal_distance_to_soma  42593 non-null  float64
 3   fw_similarity                  42593 non-null  float64
 4   me_similarity                  42593 non-null  float64
 5   post_nucleus_x                 42593 non-null  int64
 6   pre_oracle                     42593 non-null  float64
 7   pre_test_score                 42593 non-null  float64
 8   post_oracle                    42593 non-null  float64
 9   post_test_score                42593 non-null  float64
 10  nuclues_dist                   42593 non-null  float64
 11  compartment                    42593 non-null  category
 12  ID                             42593 non-null  int64
dtypes: category(1), float64(10), int64(2)
memory usage: 4.3 MB
```

5.1 pred

[ ]:
```
# linera ensemble model
preds=[]
for model in models:
    # predict on test data
    pred_test = model.predict(lb_data2.drop(columns=['ID']))
    preds.append(pred_test)
preds=np.array(preds).T # n_samples*100
lb_data2['connected']= combine_model.predict(preds)


# vote ensemble model
# preds=[]
# for model in models:
#     # predict on test data
#     pred_test = model.predict(lb_data2.drop(columns=['ID']))
#     preds.append(pred_test)
# preds=np.array(preds)
# preds=preds.sum(axis=0) # n_samples, range(0,100)
# thresh=90
# preds[preds<thresh]=0
# preds[preds>=thresh]=1
```

```
# lb_data2['connected']=preds
```

[ ]:
```
#columns should be ID, connected
submission_data = lb_data2.filter(['ID','connected'])
```

[ ]:
```
#writing csv files
import datetime
date = datetime.datetime.now().strftime("%d%H%M")
submission_data.to_csv(f'result/lgbm_linear_ensemble_{date}.csv', index=False)
```

[ ]:
```

```