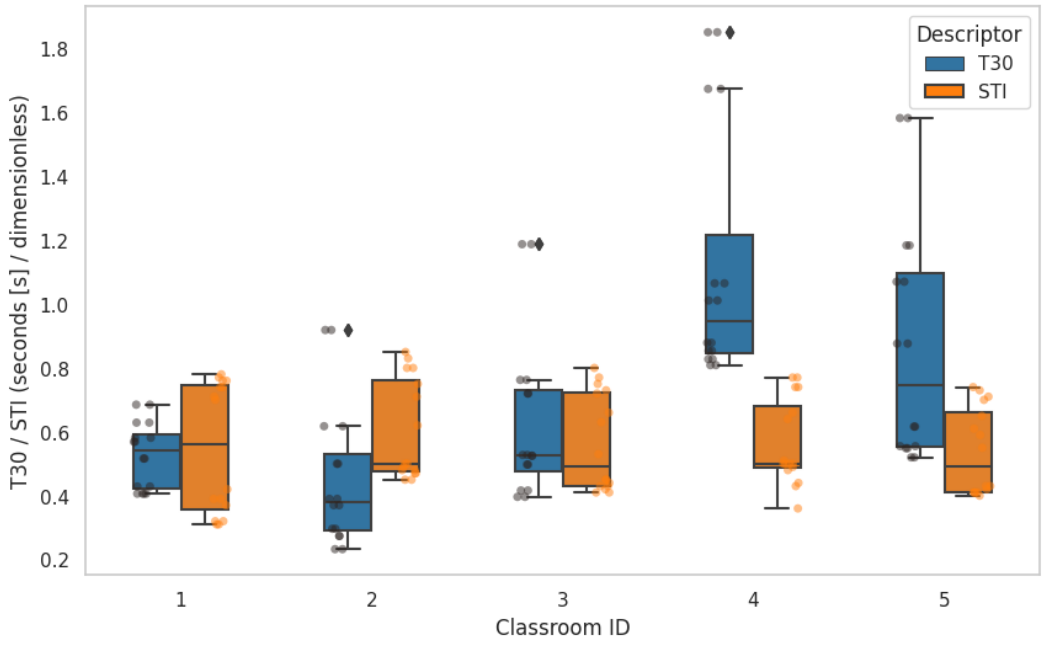
```
!pip install tensorflow==2.14.1
!pip install keras==2.14.0
!pip install eli5
!pip install shap
!pip install scikeras
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow==2.14.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow==
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow==
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.15,>=2.14->tensorflow==2.14
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboa
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<1.1,>=0.5->ten
Installing collected packages: tensorflow
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.14.0
    Uninstalling tensorflow-2.14.0:
      Successfully uninstalled tensorflow-2.14.0
Successfully installed tensorflow-2.14.1
Requirement already satisfied: keras==2.14.0 in /usr/local/lib/python3.10/dist-packages (2.14.0)
Collecting eli5
  Downloading eli5-0.13.0.tar.gz (216 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 216.2/216.2 kB 4.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: attrs>17.1.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (23.1.0)
Requirement already satisfied: jinja2>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (3.1.2)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from eli5) (1.11.4)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from eli5) (1.16.0)
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.10/dist-packages (from eli5) (1.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from eli5) (0.20.1)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.10/dist-packages (from eli5) (0.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=3.0.0->eli5) (2.1.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->eli5) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->eli5) (3.2.0)
Building wheels for collected packages: eli5
  Building wheel for eli5 (setup.py) ... done
  Created wheel for eli5: filename=eli5-0.13.0-py2.py3-none-any.whl size=107717 sha256=0d913c72241a82fce35c9f46009f2cbe2fdffd234cdc485a7ce3e31b76456581
  Stored in directory: /root/.cache/pip/wheels/b8/58/ef/2cf4c306898c2338d51540e0922c8e0d6028e07007085c0004
Successfully built eli5
Installing collected packages: eli5
Successfully installed eli5-0.13.0
Collecting shap
  Downloading shap-0.44.0-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (533 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 533.5/533.5 kB 6.9 MB/s eta 0:00:00
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)
Collecting slicer==0.0.7 (from shap)
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
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)
Installing collected packages: slicer, shap
Successfully installed shap-0.44.0 slicer-0.0.7
Collecting scikeras
  Downloading scikeras-0.12.0-py3-none-any.whl (27 kB)
```

```
%matplotlib inline

# libraries & dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
sns.set_theme(style="white")

import keras
from keras.models import Sequential
from keras.layers import Dense
from scikeras.wrappers import KerasRegressor
from keras import optimizers

# Create a History callback to record training history
from keras.callbacks import History

history_callback = History()

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score

from scipy.stats import zscore
from scipy.stats import f_oneway


import eli5
from eli5.sklearn import PermutationImportance

# libraries & dataset
from string import ascii_letters
import shap

# print the JS visualization code to the notebook
shap.initjs()
```

```
# Read CSV file into a DataFrame
raw_dataset = pd.read_csv('database.csv', sep=',')
df = raw_dataset.copy()

# Filter data for Descriptor T30 and STI
filtered_df = df[df['Descriptor'].isin(['T30', 'STI'])]

# Map Descriptor values to units
unit_mapping = {'T30': 'seconds [s]', 'STI': 'dimensionless'}
filtered_df['Descriptor'] = filtered_df['Descriptor'].map(unit_mapping)

# Define a scientific color palette
scientific_palette = ['#1f77b4', '#ff7f0e']

# Set plotting style to whitegrid
sns.set_style("whitegrid")

# Create a boxplot with data distribution
plt.figure(figsize=(10, 6))
ax = sns.boxplot(x="Classroom_id",
                 y="Value",
                 hue="Descriptor",
                 data=filtered_df,
                 palette=scientific_palette, width=0.5)

# Add strip plot for data distribution
sns.stripplot(x="Classroom_id",
              y="Value",
              hue="Descriptor",
              data=filtered_df,
              color='#ff7f0e',
              dodge=True,
              alpha=0.5)

# Set x-axis label
ax.set_xlabel("Classroom ID")

# Set y-axis label with unit information
ax.set_ylabel(f"T30 / STI ({unit_mapping['T30']} / {unit_mapping['STI']})")

# Add legend with descriptor names and set STI box to be orange
legend = ax.legend(title='Descriptor',
                   loc='upper right',
                   labels=['T30', 'STI'])

legend.legendHandles[1].set_facecolor('#ff7f0e')  # Set color for STI box

# Remove grid lines
ax.yaxis.grid(False)

# Show the plot
plt.show()
```

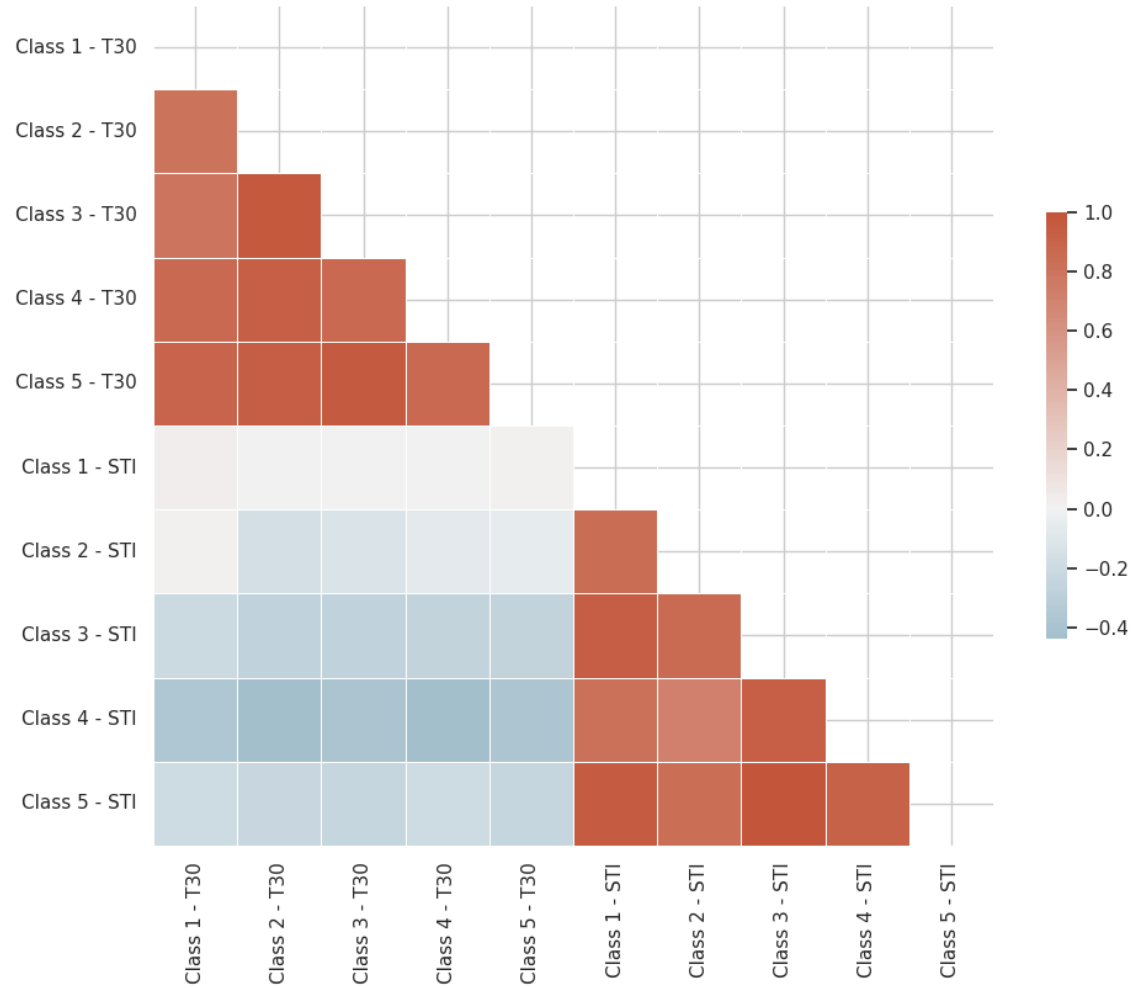
```
Setting a gradient palette using color= is deprecated and will be removed in version 0.13. Set `palette='dark:#ff7f0e'` for same effect.
The legendHandles attribute was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use legend_handles instead.
```




```
# Generate dataset
raw_dataset = pd.read_csv('/content/data_to_correlation.csv', sep=',')
d = raw_dataset.copy()
df.tail()

# Compute the correlation matrix
corr = d.corr()
```

```
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1.0, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
<Axes: >
```



```
from scipy.stats import linregress


def auto_sensitivy_analisys(raw_dataset_file, x_label, y_label):

  # Load entire dataset for training
  raw_dataset = pd.read_csv(raw_dataset_file, sep=',')
  df = raw_dataset.copy()
  df.tail()

  # Set up variables to the tranings variables
  df = df.sort_index()

  df = df[y_label + x_label]
  df = df.dropna()
  df.tail()

  # Split the data into test and training sets.
  np.random.seed(100)
  X_train, X_test, y_train, y_test = train_test_split(df[x_label],
                                                      df[y_label],
                                                      test_size=0.01)
  # Print the dimensions
  print('Training set dimensions X, y: ' + str(X_train.shape) + ' ' +str(y_train.shape))
  print('Test set dimensions X, y: ' + str(X_test.shape) + ' '+ str(y_test.shape))


  # Define regression model in Keras
  def DL_regression_model():
      model = Sequential()
      model.add(Dense(8, input_dim=4, activation='relu'))
      model.add(Dense(4, activation='relu'))
      model.add(Dense(2, activation='relu'))
      model.add(Dense(1, activation='linear'))

      # Compile model
      adam = optimizers.Adam(0.01)
      model.compile(loss='mean_squared_error',
                    optimizer=adam,
                    metrics=['mse'])

      return model

  # O parâmetro patience é o quantidade de epochs para checar as melhoras
  early_stop = keras.callbacks.EarlyStopping(monitor='val_loss',
                                             patience=5)

  estimator = KerasRegressor(build_fn=DL_regression_model,
                             validation_split = 0.1,
                             batch_size=2,
                             loss='mse',
                             epochs=20,
                             verbose=0,
                             callbacks=[early_stop,
                                        history_callback]
                             )

  history = estimator.fit(X_train, y_train)

  print(DL_regression_model)

  isPlot = False

  if isPlot == True:
   # Plot training & validation loss values

   plt.plot(history_callback.history['loss'])
   plt.plot(history_callback.history['val_loss'])
   plt.title('Model loss')
   plt.xlabel('Epoch')
   plt.ylabel('Loss')
   plt.legend(['Train', 'Validation'], loc='upper right')
   plt.show()


  isCross = False

  if isCross == True:
    # Cross validation

    kfold = KFold(n_splits=4)
    result = cross_val_score(estimator,
                             X_train.values,
                             y_train.values,
                             cv=kfold,
                             n_jobs=1)

    print("Results: %.2f (%.2f) MSE" % (result.mean(), result.std()))

    # Assuming 'result' is the array of cross-validation results
    cv_results = pd.DataFrame({'Fold': range(1, len(result) + 1), 'MSE': result})
```

```
  # Display the table
  print(cv_results)

# Fitting resuts

fitted = estimator.predict(X_train)
residuals = y_train[y_label[0]].values - fitted

# Two plots
fig, (ax1, ax2) = plt.subplots(ncols=2,figsize=(12,6))

# 1. Histogram of residuals
sns.distplot(residuals, ax=ax1)
ax1.set_title('Histogram of residuals')

# Fitted vs residuals
x1 = pd.Series(fitted.reshape(15), name='Fitted total')
x2 = pd.Series(y_train[y_label[0]], name="total values")

#sns.kdeplot(x1, n_levels=5,ax = ax2)
sns.regplot(x=x1, y=x2, scatter=False, ax = ax2)
ax2.set_title('Fitted vs actual values')
ax2.set_xlim([0,1])
ax2.set_ylim([0,1])
ax2.set_aspect('equal')

# Calculate linear DL model regression parameters
DL_slope, intercept, r_value, p_value, std_err = linregress(y_train[y_label[0]].values,
                                                            fitted.reshape(15))

# Print the slope coefficient
print(f"Slope Coefficient: {DL_slope}")

plt.show()

# Permutation model
perm = PermutationImportance(estimator,
                             random_state=8).fit(X_train,y_train)

perm_importances = eli5.explain_weights_df(perm, feature_names=X_train.columns.tolist())

# SHAP expects model functions to take a 2D numpy array as input,
# so we define a wrapper function around the original Keras predict function.
def f_wrapper(X):
    return estimator.predict(X).flatten()

X_train_summary = shap.kmeans(X_train, 10)
X_train_summary = X_train_summary.data.astype(int)

# Compute Shap values
explainer = shap.KernelExplainer(f_wrapper, X_train_summary)

# Create subplots
plt.figure(1)
# Make plot with combined shap values
X_train_sample = X_train.sample(10)
shap_values  = explainer.shap_values(X_train_sample)
shap.summary_plot(shap_values, X_train_sample)
plt.show()

print(shap_values)

column_means = shap_values.max(axis=0)

print("Shape values", column_means.sort)
print("Shape values", np.sort(column_means))

if isPlot == True:
 # Dependence plots
 shap.dependence_plot(0, shap_values, X_train_sample)
 shap.dependence_plot(1, shap_values, X_train_sample)
 shap.dependence_plot(2, shap_values, X_train_sample)
 shap.dependence_plot(3, shap_values, X_train_sample)

shape_result = np.mean(shap_values, axis=0)
print("Shap values final", shape_result)

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = regr.predict(X_train)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(y_train, y_pred))
# The coefficient of determination: 1 is perfect prediction
```

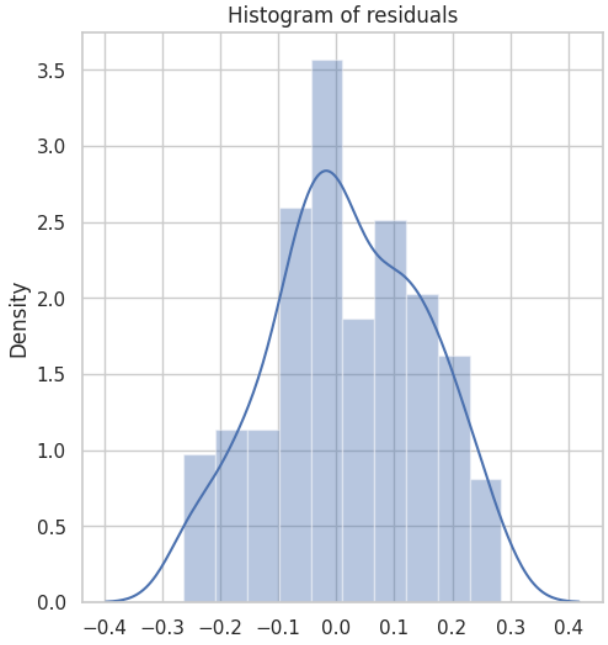
```
  print('Coefficient of determination: %.2f'
        % r2_score(y_train, y_pred))

  return y_label[0], perm_importances['weight'].values, shape_result, regr.coef_[0], DL_slope
```

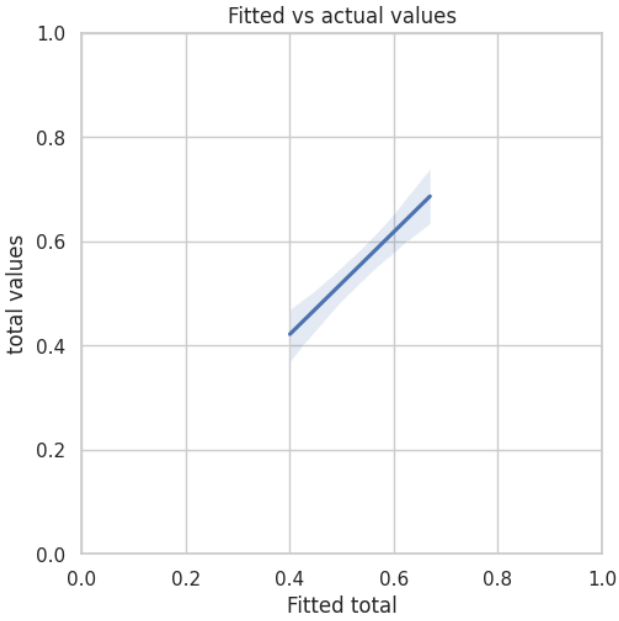
```
raw_dataset.columns.values
```

```
     array(['Class 1 - T30', 'Class 2 - T30', 'Class 3 - T30', 'Class 4 - T30',
            'Class 5 - T30', 'Class 1 - STI', 'Class 2 - STI', 'Class 3 - STI',
            'Class 4 - STI', 'Class 5 - STI'], dtype=object)
```

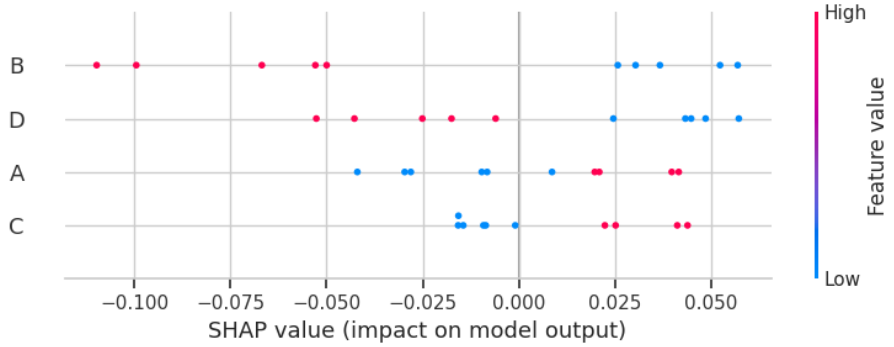
```
x_label = ['A', 'B',  'C',  'D']
raw_dataset_file = '/content/training_dataset.csv'
result = []

for class_id in raw_dataset.columns.values:
    print(f"Evaluating classroom: {class_id}")

    while True:
        partial_resul = auto_sensitivy_analisys(raw_dataset_file, x_label, [class_id])
        if partial_resul[2].mean() != 0 or partial_resul[4] > 0.9:
            result.append(partial_resul[:-1])
            break  # Exit the loop if the condition is met
```

```
Evaluating classroom: Class 1 - T30
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbba8c0d1b0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.6622036894041897
```

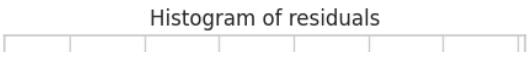


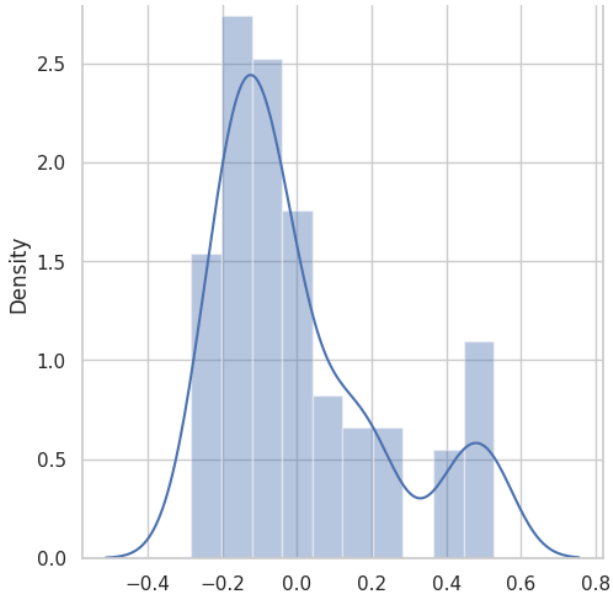


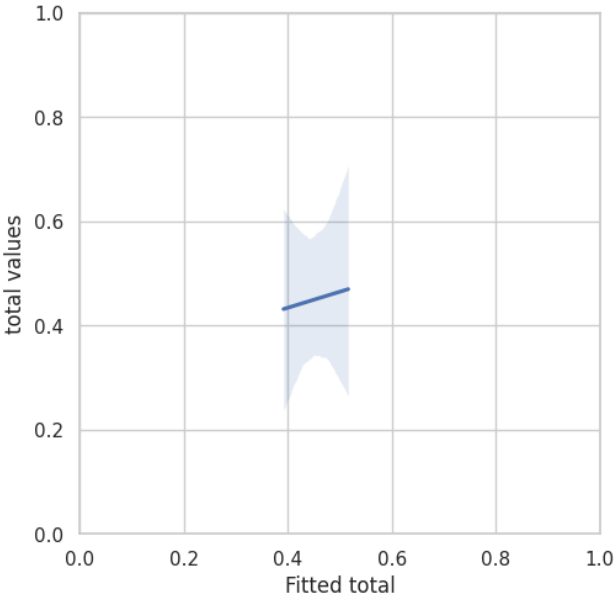
100% 10/10 [00:07<00:00, 1.38it/s]



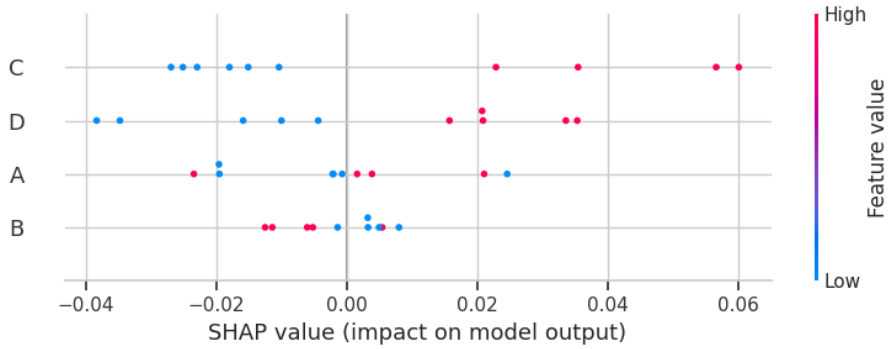

```
[[ 0.019722   -0.10978085  0.04118198 -0.00604939]
 [ 0.00861451 -0.06683198 -0.00095176  0.04854783]
 [-0.00965422 -0.05291691 -0.01572005 -0.04276023]
 [-0.0296959   0.03667073  0.02236027  0.04478509]
 [-0.00824867 -0.04998423  0.04384693 -0.01751552]
 [-0.0419967   0.02572093 -0.01445953 -0.05266192]
 [ 0.02090354 -0.09947613 -0.01579792 -0.02510101]
 [ 0.03974038  0.05230211 -0.00923568  0.04328648]
 [ 0.04155864  0.05689445  0.02514711  0.02456229]
 [-0.02810164  0.03034129 -0.00861001  0.05718156]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbba1fc0330>
Shape values [0.04155864 0.04384693 0.05689445 0.05718156]
Shap values final [ 0.00128419 -0.01770606  0.00677613  0.00742752]
Coefficients:
 [[-0.00042614 -0.08785511 -0.00839489 -0.03183239]]
Mean squared error: 0.00
Coefficient of determination: 0.89
Evaluating classroom: Class 2 - T30
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbba80e7370>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.014423736143132365
```



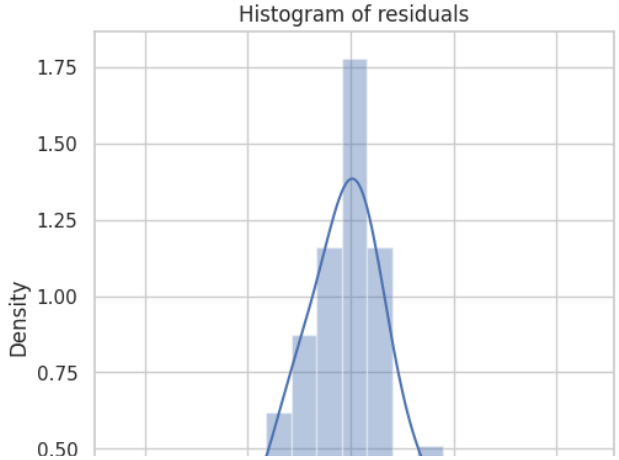


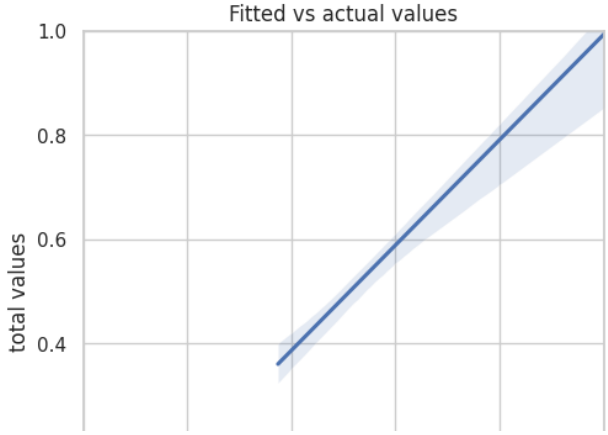


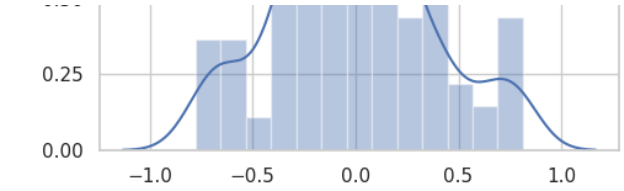
100% 10/10 [00:07<00:00, 1.04it/s]

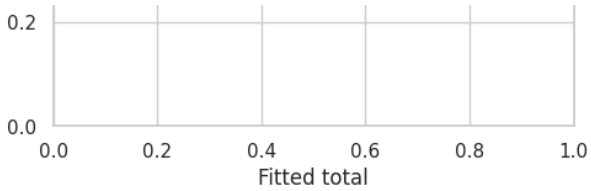



```
[[ 0.00384892 -0.00606676  0.03539174  0.03526635]
 [-0.00217039 -0.00519811 -0.0229193  -0.01000388]
 [-0.01950245 -0.01141195 -0.02509079  0.01571351]
 [ 0.02453845  0.00492649  0.05653928 -0.00439088]
 [-0.00206758  0.00541301  0.0599957   0.02070737]
 [-0.01956753  0.00798013 -0.01796611  0.02083954]
 [ 0.0210168  -0.01250738 -0.015092    0.033526  ]
 [ 0.00157728  0.00324347 -0.01038767 -0.03472476]
 [-0.02340814 -0.00141184  0.02282093 -0.03829263]
 [-0.00071396  0.00320062 -0.02690715 -0.01587119]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbba1399830>
Shape values [0.00798013 0.02453845 0.03526635 0.0599957 ]
Shap values final [-0.00164486 -0.00118323  0.00563846  0.00227694]
Coefficients:
 [[ 0.01014205 -0.1618608   0.0865483  -0.0562642 ]]
Mean squared error: 0.01
Coefficient of determination: 0.76
Evaluating classroom: Class 3 - T30
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbba158f6d0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.9447388169451713
```

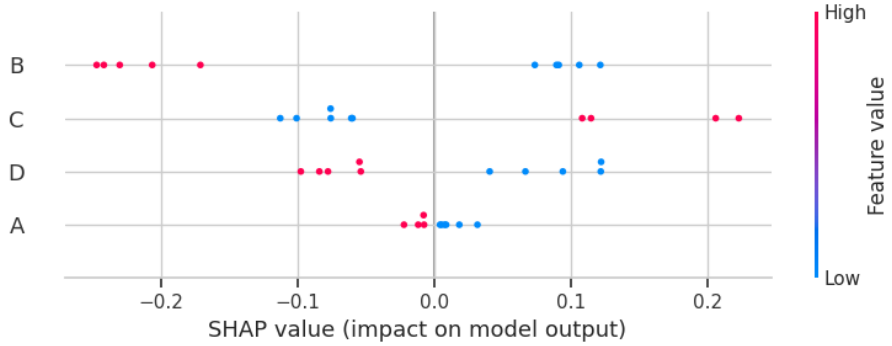




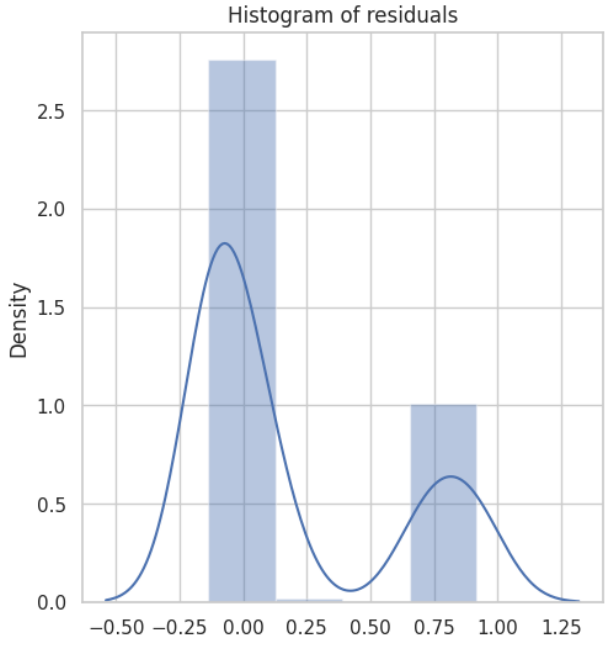


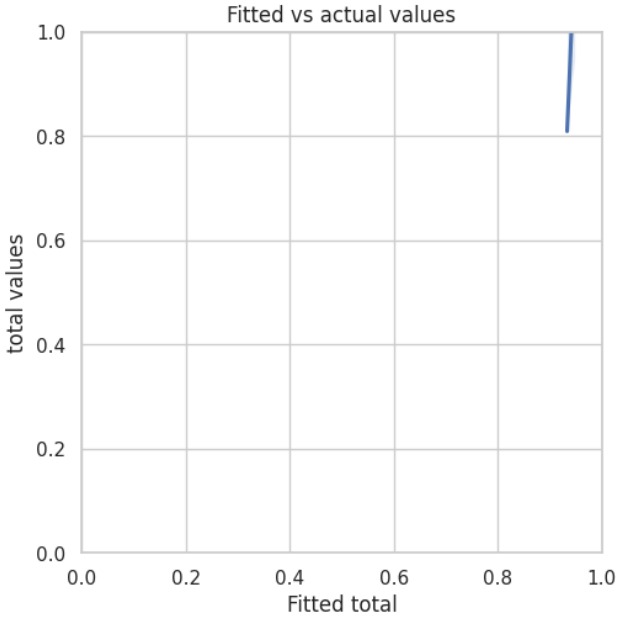


100% 10/10 [00:05<00:00, 2.70it/s]

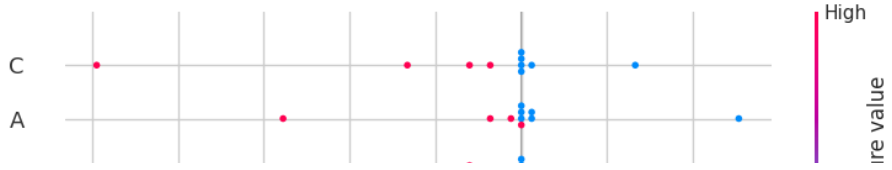


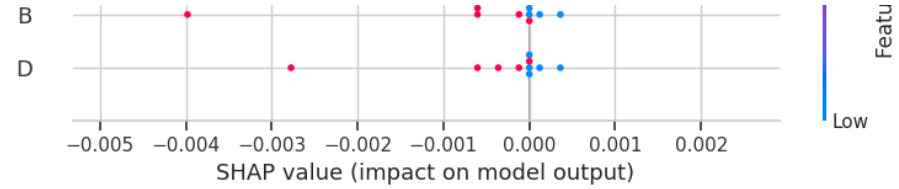

```
[[-0.01166481 -0.23013576  0.10838874 -0.09757703]
 [ 0.00761925 -0.24704509 -0.07566655  0.04066474]
 [ 0.00540331 -0.20636277 -0.06065285 -0.05368153]
 [ 0.01844169  0.12165451  0.22303528  0.12199277]
 [ 0.00878005 -0.24177059  0.11485586 -0.07769762]
 [ 0.03171065  0.07372349 -0.07580235 -0.08402329]
 [-0.00771062 -0.17110441 -0.05978905 -0.05467145]
 [-0.00736609  0.09126264 -0.10070589  0.09422029]
 [-0.02207279  0.10624949  0.20611924  0.12223438]
 [ 0.00431242  0.08954026 -0.11262582  0.06682206]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb99540150>
Shape values [0.03171065 0.12165451 0.12223438 0.22303528]
Shap values final [ 0.00274531 -0.06139882  0.01671566  0.00782833]
Coefficients:
 [[ 0.00610795 -0.17579545  0.12079545 -0.07764205]]
Mean squared error: 0.01
Coefficient of determination: 0.83
Evaluating classroom: Class 4 - T30
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb99792200>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.0019623976874297124
```



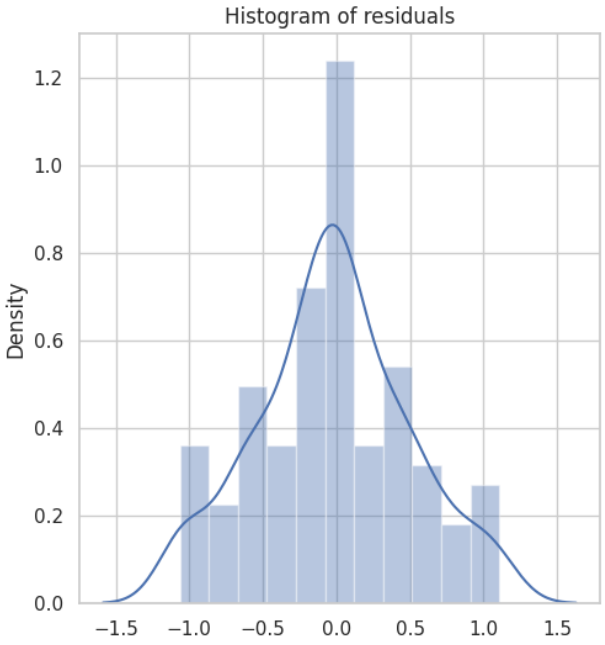


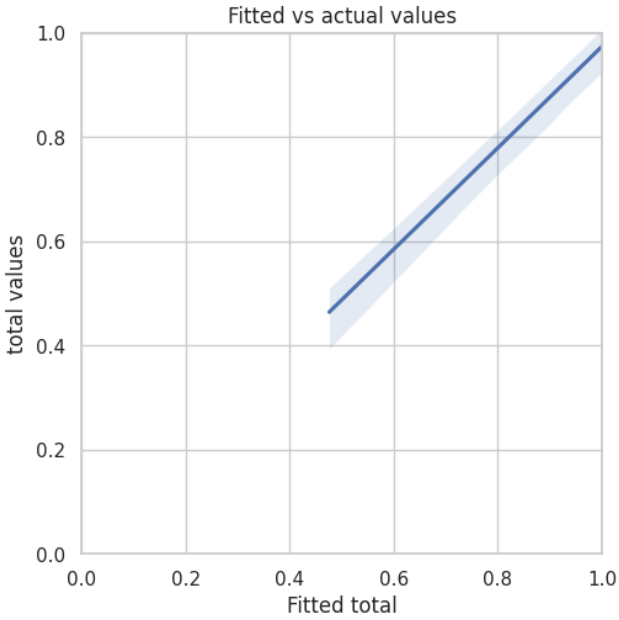
100% 10/10 [00:03<00:00, 2.25it/s]



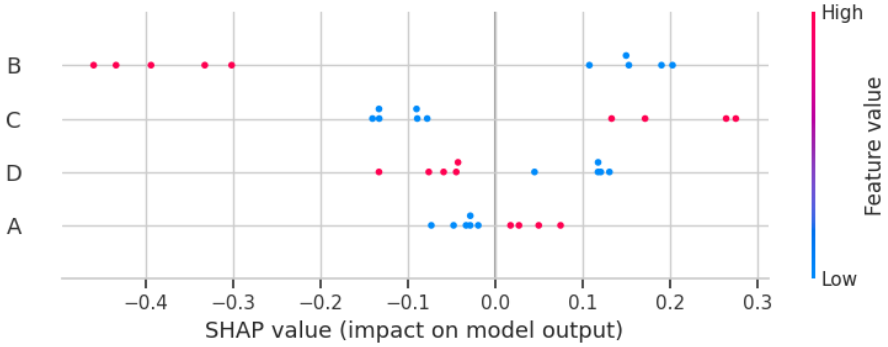



```
[[-0.00277362 -0.00397954 -0.00494428 -0.00277362]
 [ 0.          0.          0.          0.        ]
 [ 0.00012059 -0.00012059  0.00012059 -0.00012059]
 [ 0.00012059  0.00012059 -0.00036178  0.00012059]
 [ 0.00253243 -0.00060296 -0.00132651 -0.00060296]
 [ 0.          0.          0.          0.        ]
 [-0.00036178 -0.00060296  0.00132651 -0.00036178]
 [ 0.          0.          0.          0.        ]
 [-0.00012059  0.00036178 -0.00060296  0.00036178]
 [ 0.          0.          0.          0.        ]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb9935de30>
Shape values [0.00036178 0.00036178 0.00132651 0.00253243]
Shap values final [-4.82368469e-05 -4.82368469e-04 -5.78842163e-04 -3.37657928e-04]
Coefficients:
 [[ 0.015625  -0.2946875  0.0325    -0.1646875]]
Mean squared error: 0.03
Coefficient of determination: 0.78
Evaluating classroom: Class 5 - T30
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb993525f0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.9807233824690651
```



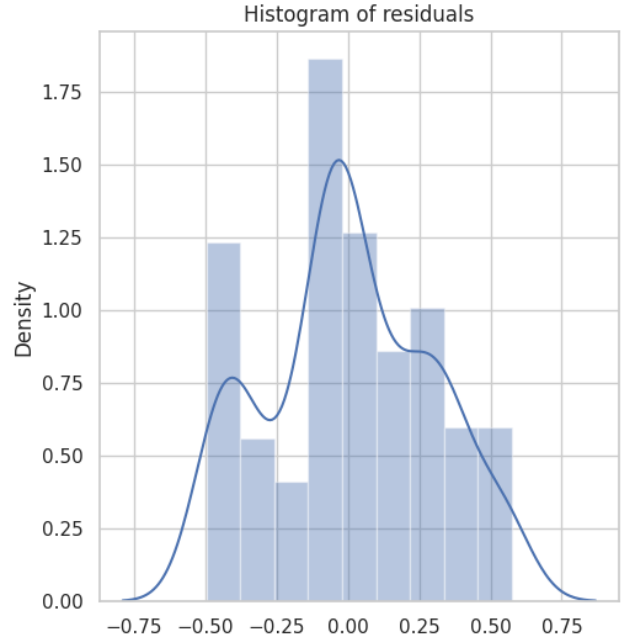


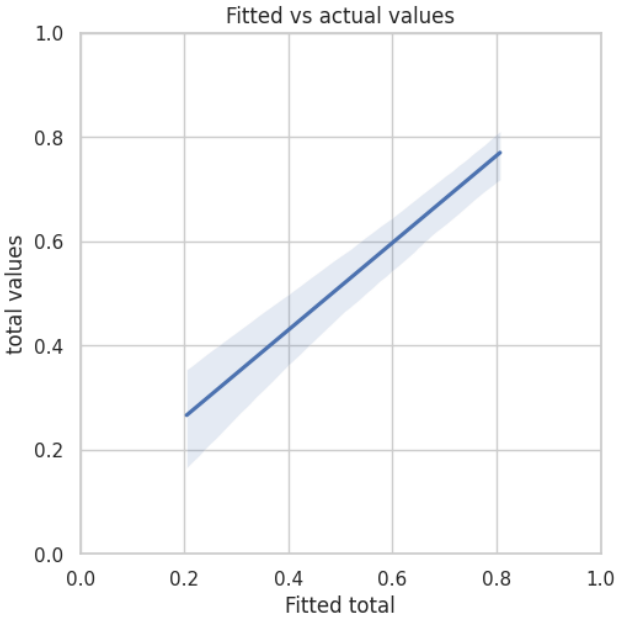

```
100%                                                 10/10 [00:02<00:00, 3.87it/s]
```



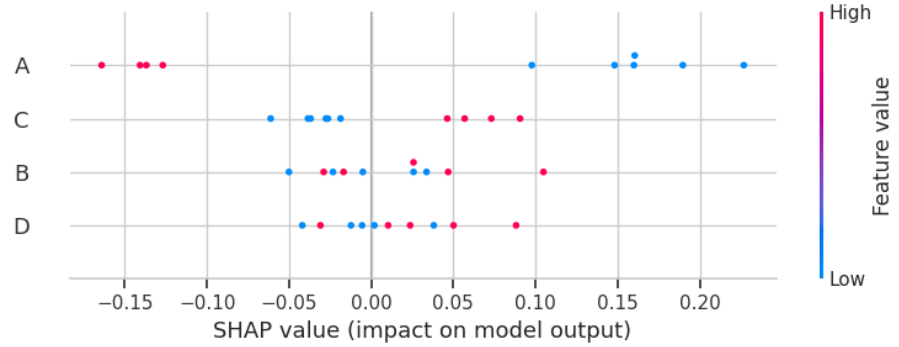

```
[[ 0.07437069 -0.39371621  0.17096942 -0.04284533]
 [-0.03362678 -0.43370508 -0.08941246  0.04461255]
 [-0.04787596 -0.33227533 -0.07807848 -0.05926594]
 [-0.01971611  0.20235889  0.27468544  0.13022538]
 [-0.07334744 -0.45928997  0.13270464 -0.07625173]
 [-0.02891998  0.10731886 -0.1331558  -0.13316874]
 [ 0.0494146  -0.30157609 -0.09024128 -0.04486102]
 [ 0.02682724  0.14929487 -0.1329292   0.11715953]
 [ 0.01721661  0.18973495  0.26363171  0.12036601]
 [-0.02881137  0.15244752 -0.14058135  0.11726814]]
```

```
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb9873dd70>
Shape values [0.07437069 0.13022538 0.20235889 0.27468544]
Shap values final [-0.00644685 -0.11194076  0.01775926  0.01732388]
Coefficients:
 [[ 0.00221591 -0.31127841  0.11752841 -0.08309659]]
Mean squared error: 0.01
Coefficient of determination: 0.89
Evaluating classroom: Class 1 - STI
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb99162ef0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.9134282591440216
```



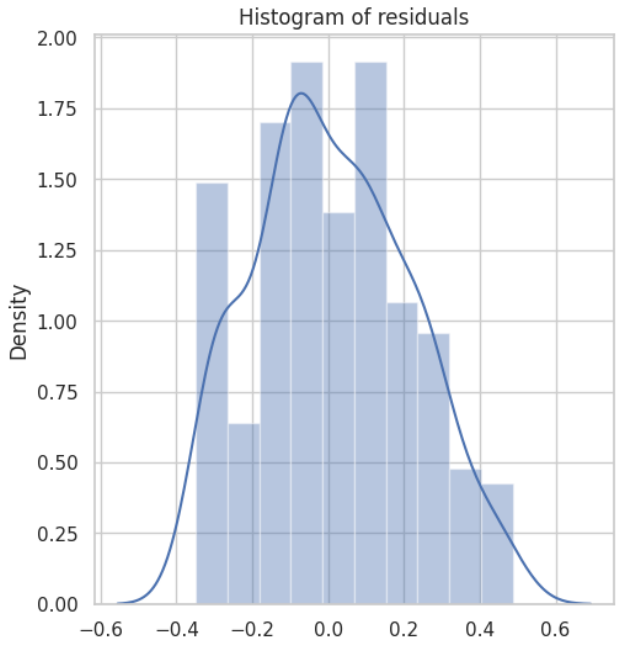


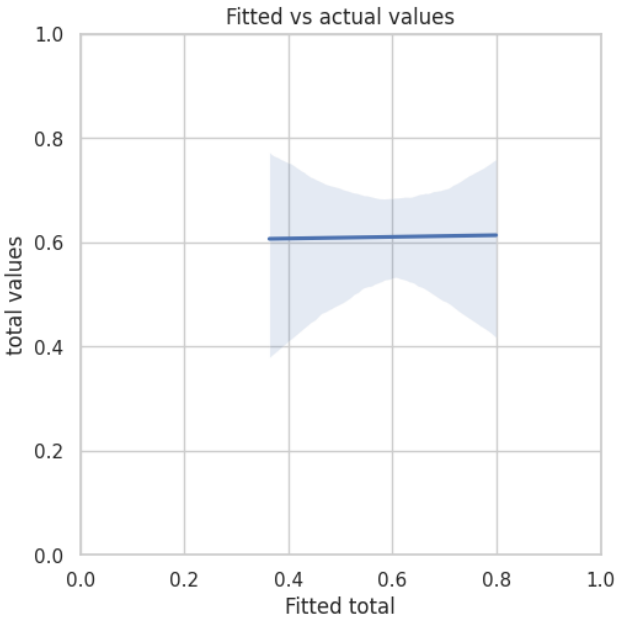

```
100%                                                 10/10 [00:02<00:00, 4.01it/s]
```



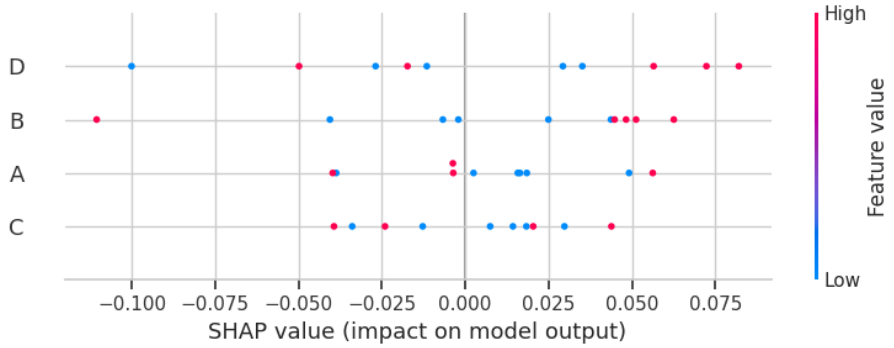

```
[[-0.13691375 -0.02894331  0.09065074  0.08823393]
 [ 0.22690888  0.04687278 -0.01866214 -0.00554652]
 [ 0.18976693  0.10489112 -0.02635139  0.01030673]
 [ 0.15997895 -0.00508536  0.07314872  0.0019685 ]
 [ 0.16047646  0.02576755  0.04630897  0.02371189]
 [ 0.09780377 -0.050104   -0.0612296  -0.03089226]
 [-0.16414224 -0.01682771 -0.0385286   0.05013207]
 [-0.12692735  0.02577768 -0.02754582 -0.01229728]
 [-0.14074355  0.03369103  0.05690824 -0.04195264]
 [ 0.1482247  -0.02325317 -0.03687579  0.03812365]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb98275a10>
Shape values [0.08823393 0.09065074 0.10489112 0.22690888]
Shap values final [0.04144328 0.01127866 0.00578233 0.01217881]
Coefficients:
 [[-0.19625 -0.0075  -0.005    0.005  ]]
Mean squared error: 0.00
Coefficient of determination: 0.97
Evaluating classroom: Class 2 - STI
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb9828ca60>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.011420352992171142
```

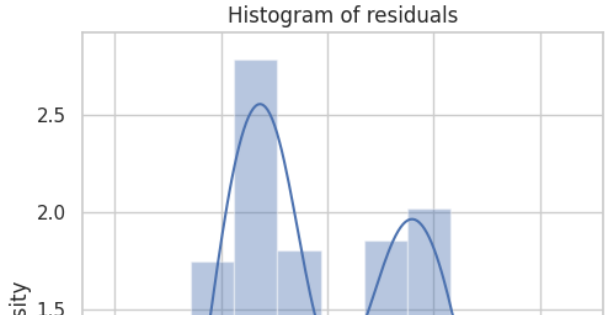


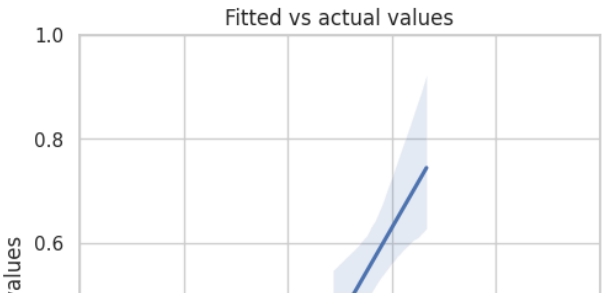


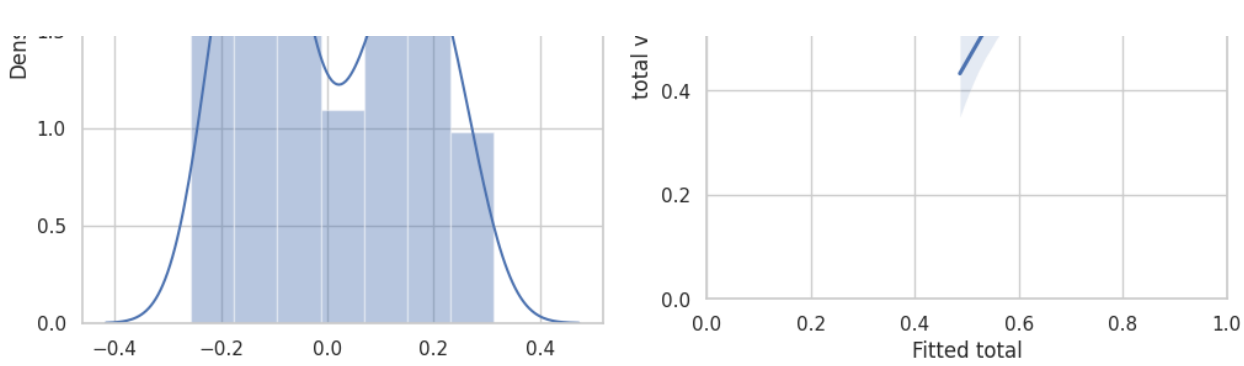

```
100%                                                10/10 [00:03<00:00, 2.67it/s]
```



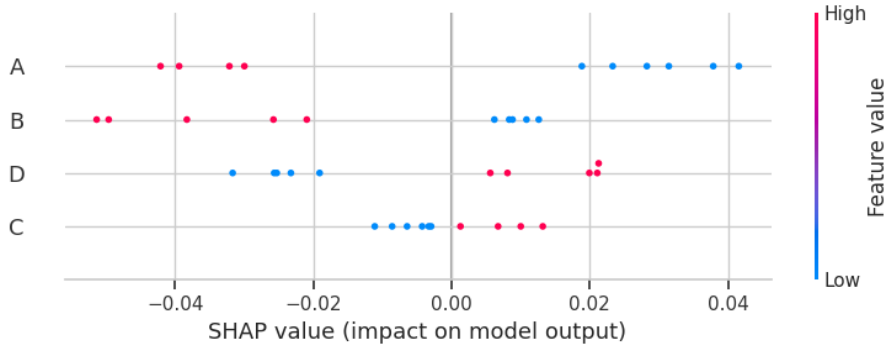

```
[[ 0.05621256  0.05121078  0.04381478  0.05647308]
 [ 0.01641    -0.11045878 -0.03386583 -0.09999849]
 [ 0.01575747  0.06252643  0.01833241  0.07230096]
 [ 0.01847075 -0.00669259 -0.02402451  0.02928081]
 [-0.03866266  0.04821499 -0.03933304 -0.01729678]
 [ 0.00249688 -0.04054043  0.01430706 -0.04980212]
 [-0.00357742  0.04477839 -0.01272988  0.08200837]
 [-0.03970475  0.02493794  0.00748855 -0.01150472]
 [-0.00368778 -0.00205248  0.02038608 -0.02685281]
 [ 0.04911785  0.04369219  0.02974203  0.03511627]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb8e6d5dd0>
Shape values [0.04381478 0.05621256 0.06252643 0.08200837]
Shap values final [0.00728329 0.01156164 0.00241177 0.00697246]
Coefficients:
 [[-0.13227273  0.00227273 -0.03977273 -0.00977273]]
Mean squared error: 0.01
Coefficient of determination: 0.75
Evaluating classroom: Class 3 - STI
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb8e5c4790>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.20049884672133783
```



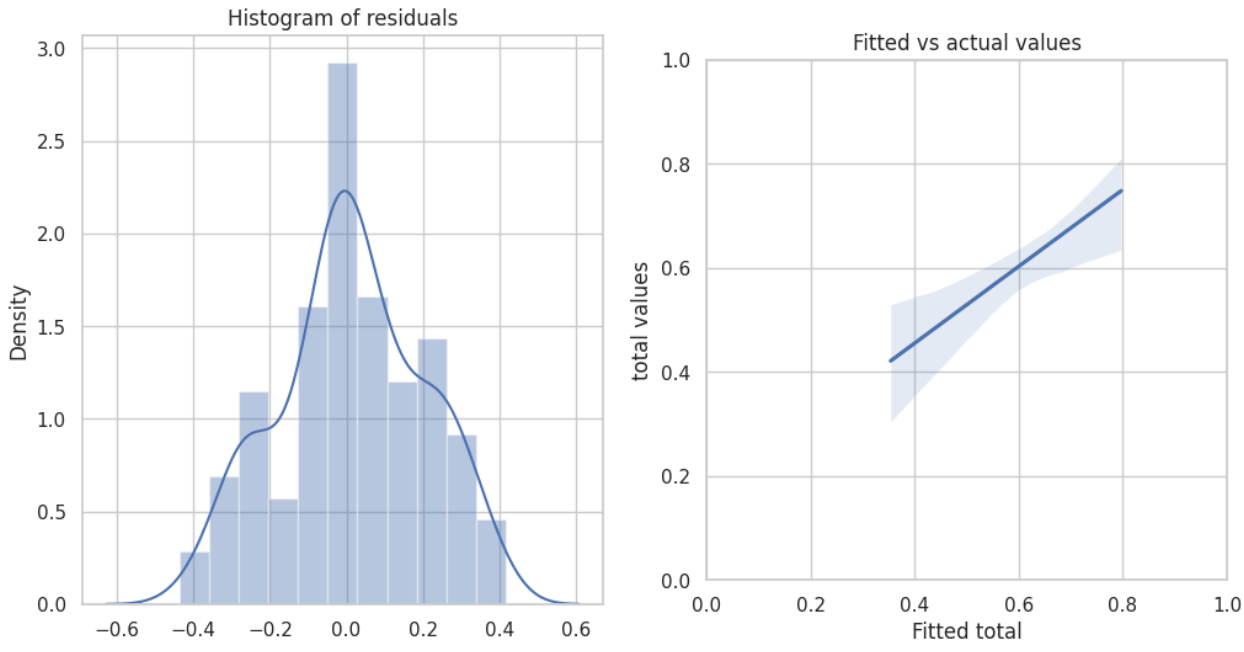




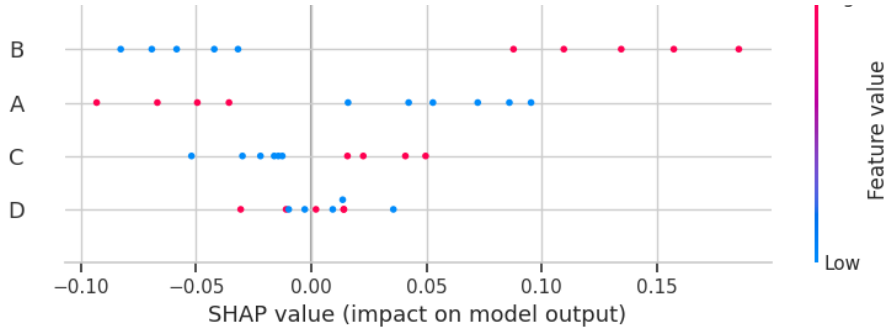
100% 10/10 [00:02<00:00, 3.79it/s]



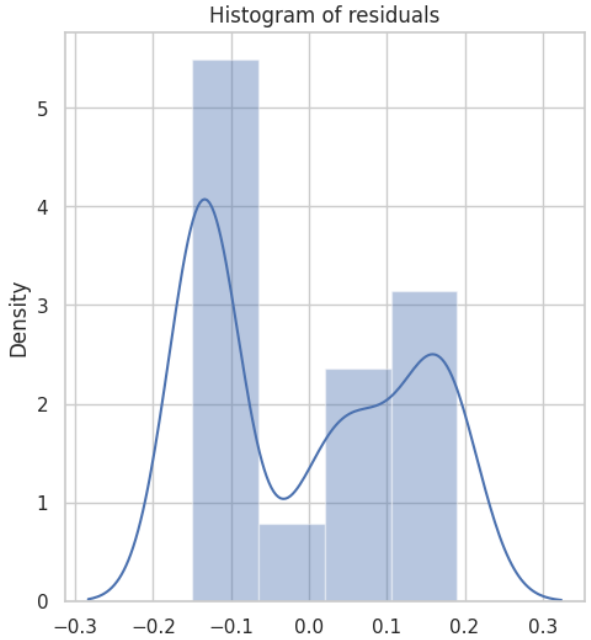

```
[[-0.03941044 -0.04959994  0.00673146  0.00809097]
 [ 0.02330528 -0.02576522 -0.00295862 -0.02324241]
 [ 0.04155235 -0.02094338 -0.00330207  0.02107984]
 [ 0.0282522   0.0126126   0.01000891 -0.02566119]
 [ 0.03785282 -0.03829279  0.00129011  0.02127253]
 [ 0.01884826  0.00833502 -0.01111348  0.01993103]
 [-0.04210067 -0.05134135 -0.00425047  0.00560053]
 [-0.03213938  0.00619294 -0.00859641 -0.0316691 ]
 [-0.02997098  0.01083621  0.01319406 -0.02528403]
 [ 0.03142991  0.00881695 -0.00643546 -0.01908745]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb98373270>
Shape values [0.0126126  0.01319406 0.02127253 0.04155235]
Shap values final [ 0.00376194 -0.0139149  -0.0005432  -0.00489693]
Coefficients:
 [[-0.13852273  0.03102273 -0.02227273  0.01522727]]
Mean squared error: 0.00
Coefficient of determination: 0.91
Evaluating classroom: Class 4 - STI
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb99569ab0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.7498570351744328
```

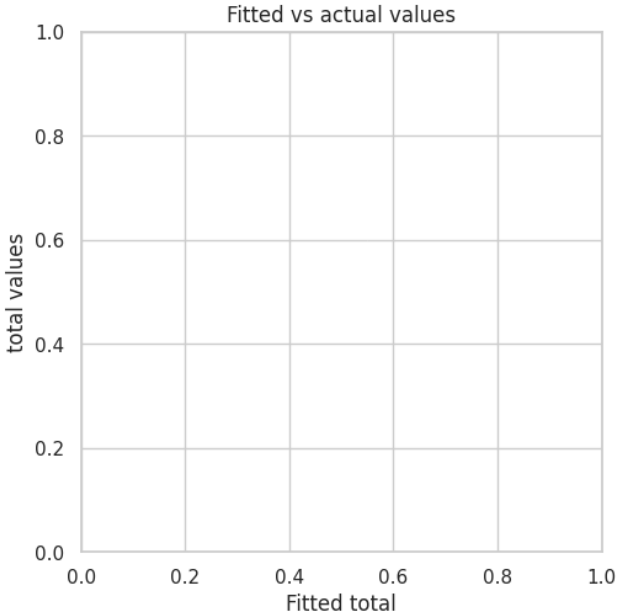


100% 10/10 [00:02<00:00, 3.45it/s]

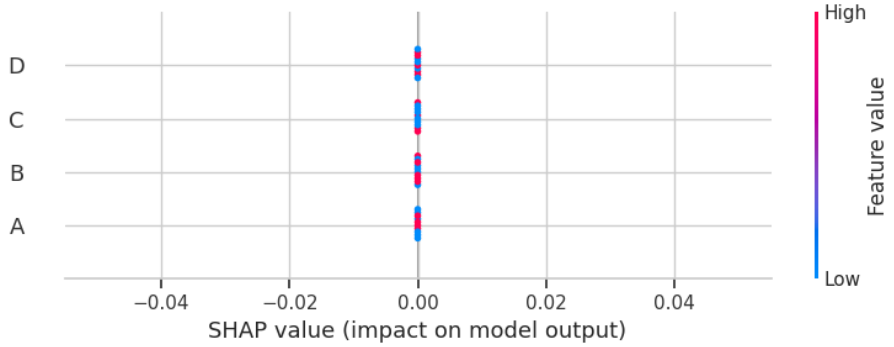



```
[[-0.09311626  0.08780894  0.01575207  0.01416793]
 [ 0.08601066  0.15737446 -0.01255307  0.01366371]
 [ 0.07228145  0.1855882  -0.02983397 -0.01090159]
 [ 0.05282663 -0.05839867  0.04088181 -0.00976177]
 [ 0.09541318  0.13454711  0.04965127  0.01411818]
 [ 0.01599625 -0.08266771 -0.05195961 -0.03060303]
 [-0.06677226  0.10963925 -0.01425745  0.00206642]
 [-0.03567212 -0.04207113 -0.01605074  0.00936459]
 [-0.04941485 -0.03177137  0.02259034 -0.00280314]
 [ 0.04229615 -0.06917922 -0.02203247  0.0356645 ]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb995bc1b0>
Shape values [0.0356645  0.04965127 0.09541318 0.1855882 ]
Shap values final [ 0.01198488  0.03908699 -0.00178118  0.00349758]
Coefficients:
 [[-0.10693182  0.04943182 -0.01693182  0.01431818]]
Mean squared error: 0.00
Coefficient of determination: 0.77
Evaluating classroom: Class 5 - STI
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbba1331630>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

Slope Coefficient: 0.0
```






```
100%                                                 10/10 [00:04<00:00, 2.94it/s]
```




```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
```

```
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Shape values <built-in method sort of numpy.ndarray object at 0x7bbb997d99b0>
Shape values [0. 0. 0. 0.]
Shap values final [0. 0. 0. 0.]
Coefficients:
 [[-0.12545455  0.03045455 -0.01795455  0.00204545]]
Mean squared error: 0.00
Coefficient of determination: 0.93
Training set dimensions X, y: (15, 4) (15, 1)
Test set dimensions X, y: (1, 4) (1, 1)
``build_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_fn`` will raise an Error instead.
<function auto_sensitivy_analisys.<locals>.DL_regression_model at 0x7bbb8e5c4af0>


`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
# result
```

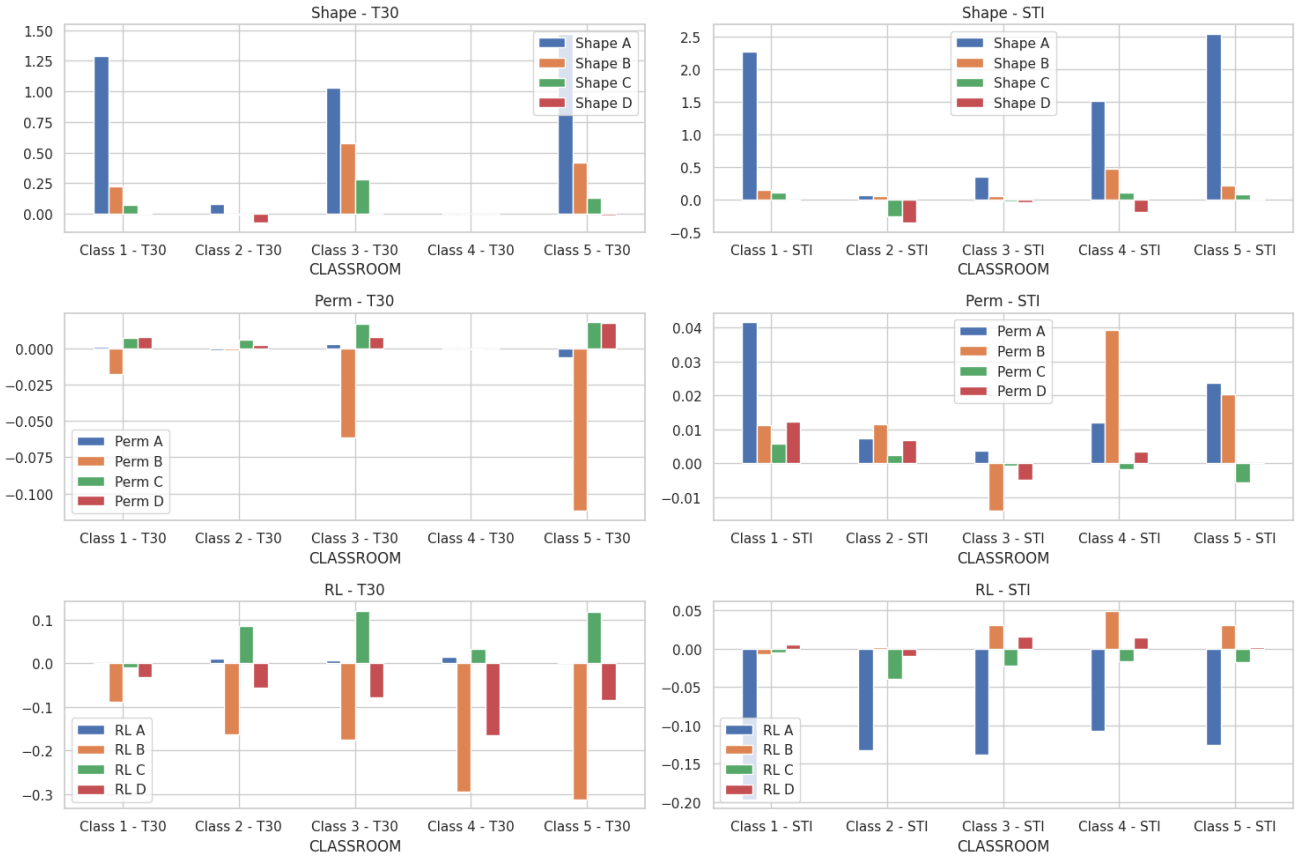
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

```
df= pd.DataFrame(result)
```

```
# Flatten the nested arrays
labels = ['A', 'B', 'C', 'D']
df_expanded = pd.concat([df[0], df[1].apply(pd.Series).rename(lambda x: f'Shape {labels[x]}', axis=1),
                         df[2].apply(pd.Series).rename(lambda x: f'Perm {labels[x]}', axis=1),
                         df[3].apply(pd.Series).rename(lambda x: f'RL {labels[x]}', axis=1)], axis=1)

# Display the expanded DataFrame
df_expanded= df_expanded.rename(columns={0: 'CLASSROOM'})
df_expanded
```

	CLASSROOM	Shape A	Shape B	Shape C	Shape D	Perm A	Perm B	Perm C	Perm D	RL A	RL B	RL C	RL D
0	Class 1 - T30	1.289115	0.225569	0.070264	0.007288	0.001284	-0.017706	0.006776	0.007428	-0.000426	-0.087855	-0.008395	-0.031832
1	Class 2 - T30	0.081982	-0.003588	-0.007129	-0.070961	-0.001645	-0.001183	0.005638	0.002277	0.010142	-0.161861	0.086548	-0.056264
2	Class 3 - T30	1.030466	0.575690	0.278094	-0.001346	0.002745	-0.061399	0.016716	0.007828	0.006108	-0.175795	0.120795	-0.077642
3	Class 4 - T30	0.000979	0.000653	-0.000131	-0.000140	-0.000048	-0.000482	-0.000579	-0.000338	0.015625	-0.294688	0.032500	-0.164688
4	Class 5 - T30	1.469469	0.415499	0.131695	-0.010879	-0.006447	-0.111941	0.017759	0.017324	0.002216	-0.311278	0.117528	-0.083097
5	Class 1 - STI	2.274096	0.150453	0.103827	-0.011500	0.041443	0.011279	0.005782	0.012179	-0.196250	-0.007500	-0.005000	0.005000
6	Class 2 - STI	0.061600	0.047049	-0.258349	-0.355131	0.007283	0.011562	0.002412	0.006972	-0.132273	0.002273	-0.039773	-0.009773
7	Class 3 - STI	0.348614	0.056079	-0.035536	-0.038198	0.003762	-0.013915	-0.000543	-0.004897	-0.138523	0.031023	-0.022273	0.015227
8	Class 4 - STI	1.515051	0.467203	0.099275	-0.193029	0.011985	0.039087	-0.001781	0.003498	-0.106932	0.049432	-0.016932	0.014318
9	Class 5 - STI	2.545768	0.208267	0.081034	0.009713	0.023609	0.020440	-0.005494	0.000127	-0.125455	0.030455	-0.017955	0.002045

```
# Filter T30 and STI classrooms
df = df_expanded
df_t30 = df[df['CLASSROOM'].str.contains('T30')]
df_sti = df[df['CLASSROOM'].str.contains('STI')]

# Select T30 and STI parameter columns for each method
shape_columns = ['Shape A', 'Shape B', 'Shape C', 'Shape D']
perm_columns = ['Perm A', 'Perm B', 'Perm C', 'Perm D']
rl_columns = ['RL A', 'RL B', 'RL C', 'RL D']

# Extract T30 and STI parameter values for each method
shape_t30 = df_t30.loc[:, ['CLASSROOM'] + shape_columns]
perm_t30 = df_t30.loc[:, ['CLASSROOM'] + perm_columns]
rl_t30 = df_t30.loc[:, ['CLASSROOM'] + rl_columns]

shape_sti = df_sti.loc[:, ['CLASSROOM'] + shape_columns]
perm_sti = df_sti.loc[:, ['CLASSROOM'] + perm_columns]
rl_sti = df_sti.loc[:, ['CLASSROOM'] + rl_columns]

# Set up the figure and axis with subplots
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 10))

# Plot bar plots for Shape, Perm, RL
shape_t30.set_index('CLASSROOM').plot(kind='bar', ax=axes[0, 0], rot=0, title='Shape - T30')
perm_t30.set_index('CLASSROOM').plot(kind='bar', ax=axes[1, 0], rot=0, title='Perm - T30')
rl_t30.set_index('CLASSROOM').plot(kind='bar', ax=axes[2, 0], rot=0, title='RL - T30')

shape_sti.set_index('CLASSROOM').plot(kind='bar', ax=axes[0, 1], rot=0, title='Shape - STI')
perm_sti.set_index('CLASSROOM').plot(kind='bar', ax=axes[1, 1], rot=0, title='Perm - STI')
rl_sti.set_index('CLASSROOM').plot(kind='bar', ax=axes[2, 1], rot=0, title='RL - STI')

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```



```
# Extract columns for each group
shape_columns = ['Shape A', 'Shape B', 'Shape C', 'Shape D']
perm_columns = ['Perm A', 'Perm B', 'Perm C', 'Perm D']
rl_columns = ['RL A', 'RL B', 'RL C', 'RL D']

# Create new DataFrames for each group
df_shape = df_expanded[shape_columns].apply(zscore, axis=1)
df_perm = df_expanded[perm_columns].apply(zscore, axis=1)
df_rl = df_expanded[rl_columns].apply(zscore, axis=1)

# Rename the columns with the respective group and z-score
df_shape.columns = [f'Shape_{i}_zscore' for i in range(len(df_shape.columns))]
df_perm.columns = [f'Perm_{i}_zscore' for i in range(len(df_perm.columns))]
df_rl.columns = [f'RL_{i}_zscore' for i in range(len(df_rl.columns))]

# Concatenate the DataFrames
df_zscores = pd.concat([df_shape, df_perm, df_rl], axis=1)

# Display the new DataFrame with z-scores
df_zscores
```

	Shape_0_zscore	Shape_1_zscore	Shape_2_zscore	Shape_3_zscore	Perm_0_zscore	Perm_1_zscore	Perm_2_zscore	Perm_3_zscore	RL_0_zscore
0	1.711762	-0.331362	-0.629710	-0.750690	0.180519	-1.683853	0.719692	0.783642	0.927395
1	1.506721	-0.067401	-0.132537	-1.306783	-0.991530	-0.834599	1.484440	0.341689	0.444075
2	1.464548	0.274637	-0.504018	-1.235167	0.364433	-1.709273	0.816079	0.528762	0.346127
3	1.305070	0.638569	-0.962896	-0.980742	1.565016	-0.601929	-1.083473	0.120386	0.876825
4	1.670217	-0.148293	-0.637964	-0.883960	0.268726	-1.702788	0.721099	0.712963	0.451045
5	1.728723	-0.503171	-0.552173	-0.673379	1.705133	-0.458487	-0.852723	-0.393923	-1.729366
6	1.021703	0.942544	-0.718871	-1.245375	0.069805	1.391261	-1.434864	-0.026202	-1.657416
7	1.681411	-0.168607	-0.747986	-0.764818	1.170673	-1.530794	0.512740	-0.152620	-1.656659
8	1.614484	-0.007620	-0.577184	-1.029680	-0.077044	1.645517	-0.951991	-0.616482	-1.583941
9	1.728159	-0.473757	-0.593610	-0.660793	1.109515	0.857226	-1.207094	-0.759647	-1.656771

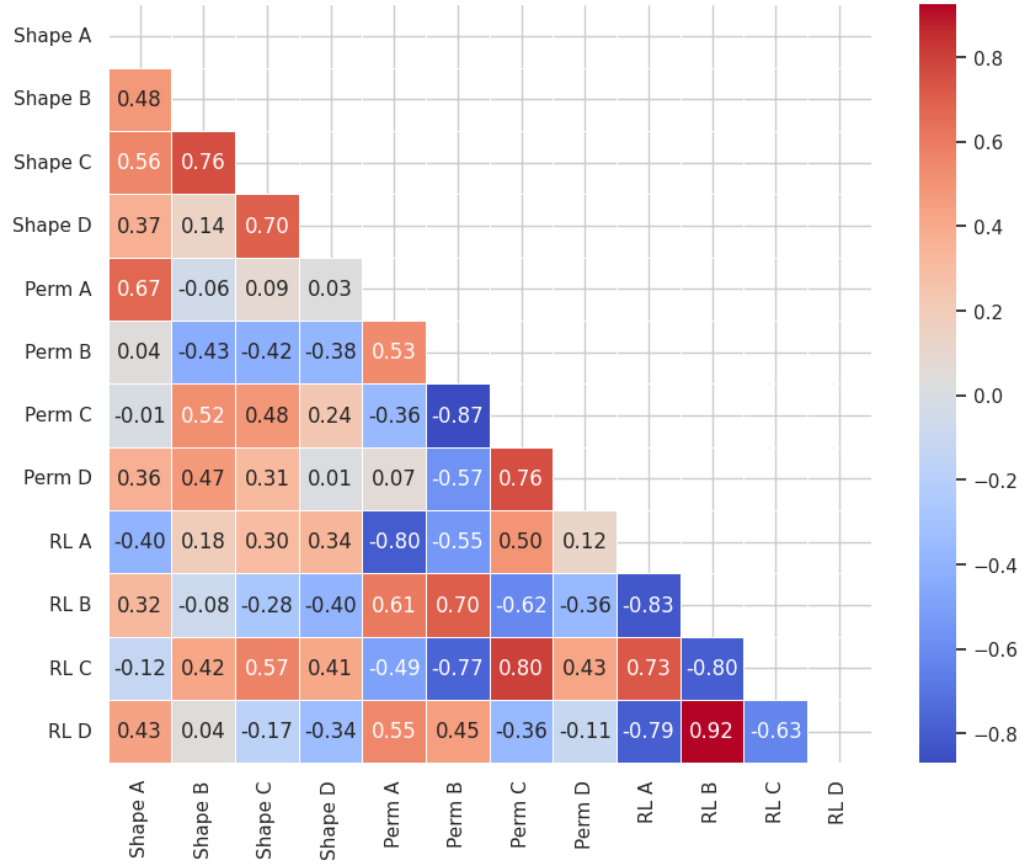
```
def perform_anova_and_decision(dataframe, group_columns, significance_level=0.05):
    """
    Perform one-way ANOVA on the specified groups in the dataframe.

    Parameters:
    - dataframe: pandas DataFrame
        The DataFrame containing the z-scores for different groups.
    - group_columns: list
        List of column names representing the groups.
    - significance_level: float, optional (default=0.05)
        The significance level for the hypothesis test.

    Returns:
    - result: str
        A string indicating whether to accept or reject the null hypothesis.
    """
    # Perform ANOVA for the specified groups
    anova_result = f_oneway(*[dataframe[col] for col in group_columns])

    # Check the p-value against the significance level
    p_value = anova_result.pvalue

    if p_value < significance_level:
        return f"Reject the null hypothesis. (p-value: {p_value:.4f})"
    else:
        return f"Accept the null hypothesis. (p-value: {p_value:.4f})"

# Example usage:
# Assuming df_zscores has been created in the previous code
shape_columns = [f'Shape_{i}_zscore' for i in range(4)]
perm_columns = [f'Perm_{i}_zscore' for i in range(4)]
rl_columns = [f'RL_{i}_zscore' for i in range(4)]

print("ANOVA Decision for Shape:", perform_anova_and_decision(df_zscores, shape_columns))
print("ANOVA Decision for Perm:", perform_anova_and_decision(df_zscores, perm_columns))
print("ANOVA Decision for RL:", perform_anova_and_decision(df_zscores, rl_columns))
```

```
ANOVA Decision for Shape: Reject the null hypothesis. (p-value: 0.0000)
ANOVA Decision for Perm: Accept the null hypothesis. (p-value: 0.1650)
ANOVA Decision for RL: Reject the null hypothesis. (p-value: 0.0418)
```

```
# In ANOVA, the null hypothesis is that the means of the groups are equal
```

```
# Calculate the correlation matrix
correlation_matrix = df_expanded.corr()

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a heatmap using seaborn with the lower triangular mask
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5, mask=mask)

# Show the plot
plt.show()
```

```
The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid column
```




```
!pip freeze
```

```
moviepy==1.0.3
mpmath==1.3.0
msgpack==1.0.7
multidict==6.0.4
multipledispatch==1.0.0
multitasking==0.0.11
murmurhash==1.0.10
music21==9.1.0
natsort==8.4.0
nbclassic==1.0.0
nbclient==0.9.0
nbconvert==6.5.4
nbformat==5.9.2
nest-asyncio==1.5.8
networkx==3.2.1
nibabel==4.0.2
nltk==3.8.1
notebook==6.5.5
notebook_shim==0.2.3
```

```
import tensorflow as tf
import datetime, os
```

```
# Load entire dataset for training

x_label = ['A', 'B',    'C',    'D']
raw_dataset_file = '/content/training_dataset.csv'
y_label = ['Class 1 - T30']


raw_dataset = pd.read_csv(raw_dataset_file, sep=',')
df = raw_dataset.copy()
df.tail()

# Set up variables to the tranings variables
df = df.sort_index()

df = df[y_label + x_label]
df = df.dropna()
df.tail()

# Split the data into test and training sets.
np.random.seed(100)
X_train, X_test, y_train, y_test = train_test_split(df[x_label],
                                                    df[y_label],
                                                    test_size=0.01)
# Print the dimensions
print('Training set dimensions X, y: ' + str(X_train.shape) + ' ' +str(y_train.shape))
print('Test set dimensions X, y: ' + str(X_test.shape) + ' '+ str(y_test.shape))
```

```
Training set dimensions X, y: (15, 4) (15, 1)
```