In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
```

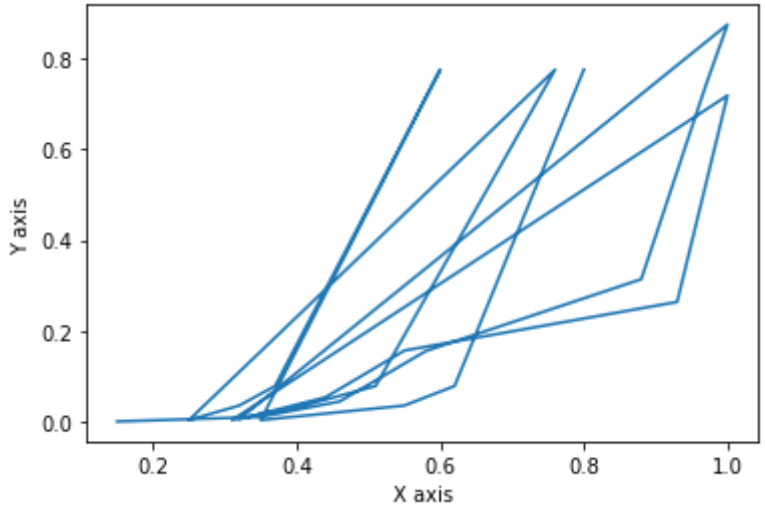
In [2]:
```
data1 = pd.read_csv('D:/class/Project/ML/sensitivity analysis/datset-1.csv')
data2 = pd.read_csv('D:/class/Project/ML/sensitivity analysis/ref dataset.csv')
```

In [3]:
```
data1.columns
```

Out[3]:
```
Index(['coat', 'time', 'in-vitro', 'in-vivo'], dtype='object')
```

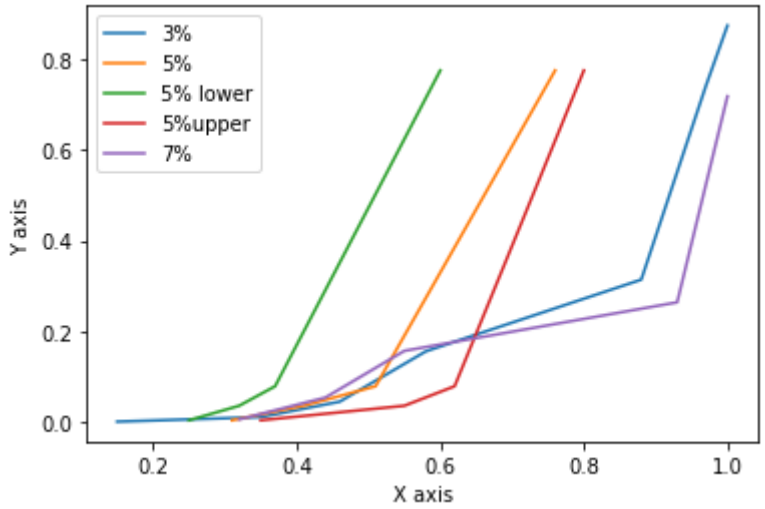
In [4]:
```
#graph for dataset 1
```

In [5]:
```
import matplotlib.pyplot as plt
x = data1['in-vitro']
y = data1['in-vivo']
plt.plot(x,y)
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```



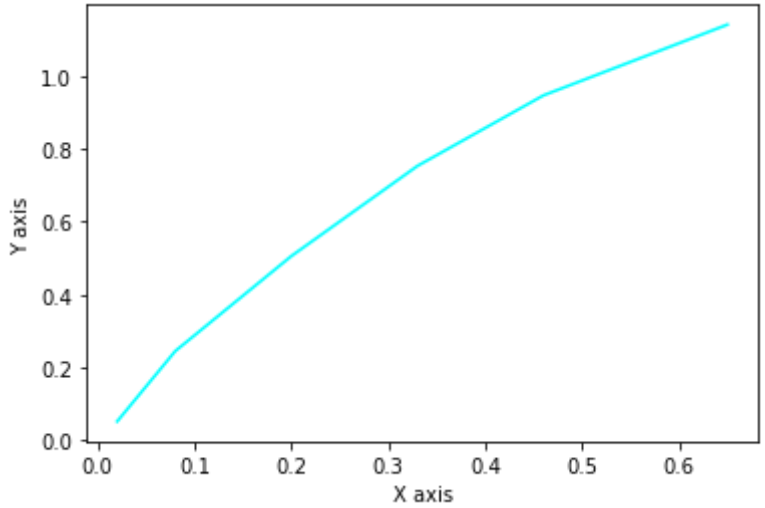
In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt

# create a dataframe with x, y, and coat columns
x = data1['in-vitro']
y = data1['in-vivo']
coat1 = data1['coat']
data_dict = {'x': x, 'y': y, 'coat': coat1}
data = pd.DataFrame(data_dict)
print(data)
# group the data by coat type
grouped_data = data.groupby('coat')

# plot each group separately
for coat_type, group in grouped_data:
    plt.plot(group['x'], group['y'], label=coat_type)

#set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# add a legend
plt.legend()

# show the graph
plt.show()
```

	x	y	coat
0	0.15	0.001	3%
1	0.35	0.011	3%
2	0.46	0.045	3%
3	0.58	0.156	3%
4	0.88	0.314	3%
5	0.97	0.741	3%
6	1.00	0.874	3%
7	0.32	0.006	7%
8	0.44	0.054	7%
9	0.55	0.157	7%
10	0.93	0.264	7%
11	1.00	0.718	7%
12	0.31	0.004	5%
13	0.41	0.036	5%
14	0.51	0.079	5%
15	0.76	0.775	5%
16	0.25	0.004	5% lower
17	0.32	0.036	5% lower
18	0.37	0.079	5% lower
19	0.60	0.775	5% lower
20	0.35	0.004	5%upper
21	0.55	0.036	5%upper
22	0.62	0.079	5%upper
23	0.80	0.775	5%upper



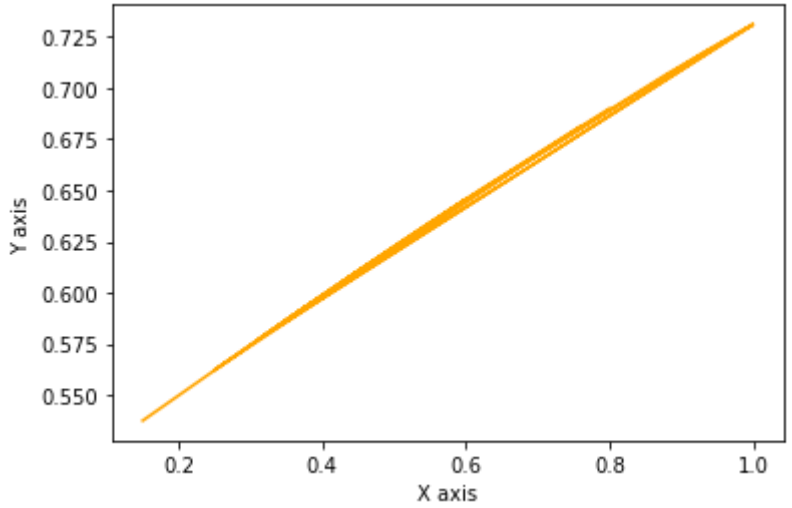
In [7]:
```
#graph for dataset-2
```

In [8]:
```
import matplotlib.pyplot as plt
x = data2['in-vitro']
y = data2['in-vivo']
plt.plot(x,y, color='cyan')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.show()
```



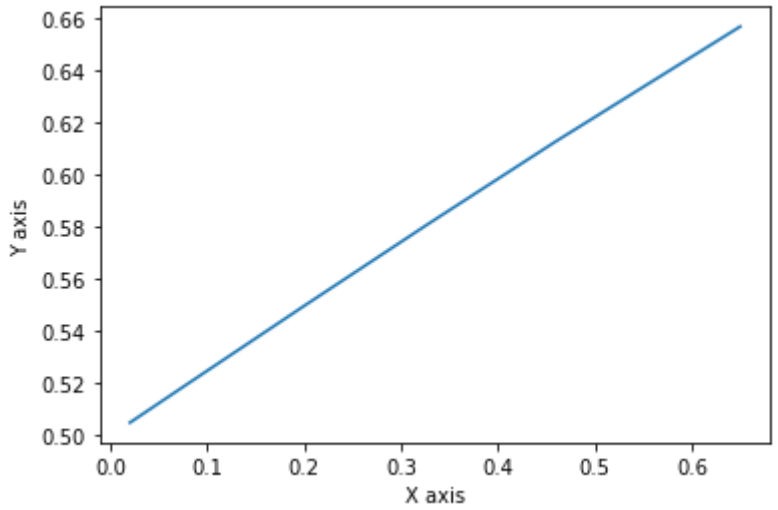
In [9]:
```
#sigmoid grapgh for dataset-1 using in-vitro
```

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt

# define the sigmoid function
x = data1['in-vitro']
def sigmoid_func(x, a, b):
    return 1 / (1 + np.exp(-(x-a)/b))

# calculate y values using the sigmoid function
y = sigmoid_func(x, 0, 1)

# plot the graph
plt.plot(x, y, color = "orange")

# set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# show the graph
plt.show()
```



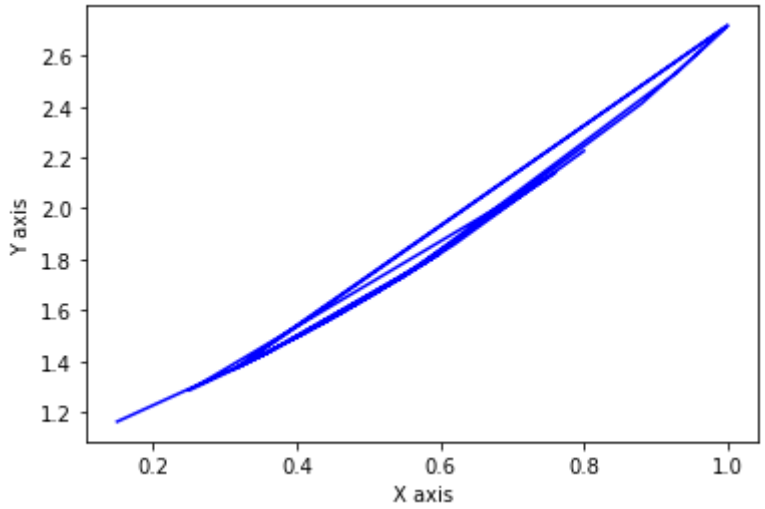
In [11]:
```
#sigmoid graph for dataset-1 using in-vitro
```

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt

# define the sigmoid function
x = data2['in-vitro']
def sigmoid_func(x, a, b):
    return 1 / (1 + np.exp(-(x-a)/b))

# calculate y values using the sigmoid function
y = sigmoid_func(x, 0, 1)

# plot the graph
plt.plot(x, y)

# set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# show the graph
plt.show()
```



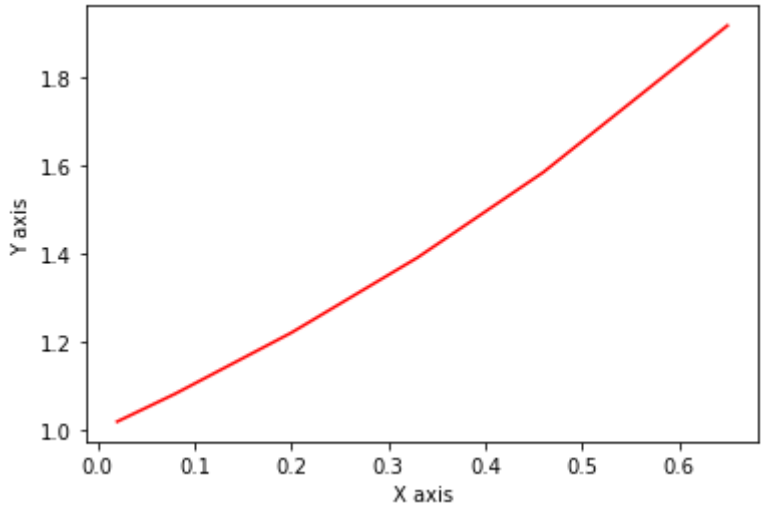
In [13]:
```
#expotential graph for dataset-1 using x
```

In [14]:
```
x = data1['in-vitro']
def exp_func(x):
    return np.exp(x)

# calculate y values using the exponential function
y = exp_func(x)

# plot the graph
plt.plot(x, y, color = "blue")

# set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# show the graph
plt.show()
```



In [15]:
```
#expotential graph for dataset-2 using x
```

In [16]:
```
x = data2['in-vitro']
def exp_func(x):
    return np.exp(x)

# calculate y values using the exponential function
y = exp_func(x)

# plot the graph
plt.plot(x, y, color = "red")

# set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# show the graph
plt.show()
```



In [17]:
```
#for Data-1 training and spliting
```

In [18]:
```
from sklearn.model_selection import train_test_split
X = data1[['in-vitro']]
y = data1[['in-vivo']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
C:\Users\HP\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is
required for this version of SciPy (detected version 1.24.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"

(19, 1) (5, 1) (19, 1) (5, 1)
```

In [19]:
```
#testing
```

In [20]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred = lin_reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.011853761559354989
R-squared: 0.8432667396266914
```

In [ ]:
```
#random forest for dataset-1
```

In [21]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf_params = {"max_depth":[100,200,300],
             "max_features":[1,2,3],
             "n_estimators":[100,150,200],
             "min_samples_split":[3,4,5]}
rf_model = RandomForestRegressor(random_state = 24)
gs = GridSearchCV(rf_model,rf_params,cv=5,n_jobs=-1,verbose=2,scoring='roc_auc')
#with np.errstate(divide='ignore', invalid='ignore'):
gs.fit(X_train,y_train.values.ravel())
print(str(gs.best_params_))
```

```
Fitting 5 folds for each of 81 candidates, totalling 405 fits
{'max_depth': 100, 'max_features': 1, 'min_samples_split': 3, 'n_estimators': 100}
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:
270 fits failed out of a total of 405.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
270 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 450, in fit
    trees = Parallel(
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 1043, in __call__
    if self.dispatch_one_batch(iterator):
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 861, in dispatch_one_batch
    self._dispatch(tasks)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 779, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\_parallel_backends.py", line 208, in apply_async
    result = ImmediateResult(func)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\_parallel_backends.py", line 572, in __init__
    self.results = batch()
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in __call__
    return [func(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in <listcomp>
    return [func(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\fixes.py", line 216, in __call__
    return self.function(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 1315, in fit
    super().fit(
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning: One or more of the test
scores are non-finite: [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
```

```
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan]
  warnings.warn(
```

In [22]:
```
rf_tune = RandomForestRegressor(max_depth= 100, max_features= 1, min_samples_split= 3, n_estimators= 100,random_state
rf_tune.fit(X_train,y_train.values.ravel())
y_pred = rf_tune.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.0009580374568277777
R-squared: 0.9873326004225328
```

In [24]:
```
#for data 2 training and spliting
```

In [50]:
```
from sklearn.model_selection import train_test_split
s = data2[['in-vitro']]
t = data2[['in-vivo']]
print(s,t)
s_train, s_test, t_train, t_test = train_test_split(s, t, test_size=0.2, random_state=0)
print(s_train.shape, s_test.shape, t_train.shape, t_test.shape)
```

```
   in-vitro
0      0.02
1      0.08
2      0.20
3      0.33
4      0.46
5      0.65     in-vivo
0  0.052223
1  0.245313
2  0.506252
3  0.753220
4  0.946752
5  1.141000
(4, 1) (2, 1) (4, 1) (2, 1)
```

In [51]:
```
#testing for refrence dataset
```

In [52]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
lin_reg = LinearRegression()
lin_reg.fit(s_train, t_train)
t_pred = lin_reg.predict(s_test)
mse = mean_squared_error(t_test, t_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(t_test, t_pred)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.02537232308258872
R-squared: 0.748106162966848
```

In [30]:
```
#random forest for Ref dataset
```

In [40]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
rf_params = {"max_depth":[100,200,300],
             "max_features":[1,2,3],
             "n_estimators":[100,150,200],
             "min_samples_split":[2,3,4]}
rf_model = RandomForestRegressor(random_state = 24)
gs = GridSearchCV(rf_model,rf_params,cv=3,n_jobs=-1,verbose=2,scoring='roc_auc')
#with np.errstate(divide='ignore', invalid='ignore'):
gs.fit(s_train,t_train)
print(str(gs.best_params_))
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
{'max_depth': 100, 'max_features': 1, 'min_samples_split': 2, 'n_estimators': 100}
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFailedWarning:
162 fits failed out of a total of 243.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
162 fits failed with the following error:
Traceback (most recent call last):
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 450, in fit
    trees = Parallel(
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 1043, in __call__
    if self.dispatch_one_batch(iterator):
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 861, in dispatch_one_batch
    self._dispatch(tasks)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 779, in _dispatch
    job = self._backend.apply_async(batch, callback=cb)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\_parallel_backends.py", line 208, in apply_async
    result = ImmediateResult(func)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\_parallel_backends.py", line 572, in __init__
    self.results = batch()
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in __call__
    return [func(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\joblib\parallel.py", line 262, in <listcomp>
    return [func(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\utils\fixes.py", line 216, in __call__
    return self.function(*args, **kwargs)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py", line 185, in _parallel_build_trees
    tree.fit(X, y, sample_weight=curr_sample_weight, check_input=False)
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 1315, in fit
    super().fit(
  File "C:\Users\HP\anaconda3\lib\site-packages\sklearn\tree\_classes.py", line 308, in fit
    raise ValueError("max_features must be in (0, n_features]")
ValueError: max_features must be in (0, n_features]

  warnings.warn(some_fits_failed_message, FitFailedWarning)
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:969: UserWarning: One or more of the test
scores are non-finite: [nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
```

```
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan]
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:926: DataConversionWarning: A column-vect
or y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel
().
  self.best_estimator_.fit(X, y, **fit_params)
```

In [41]:
```
rf_tune = RandomForestRegressor(max_depth= 100, max_features= 1, min_samples_split= 2, n_estimators= 100,random_state
rf_tune.fit(X_train,y_train.values.ravel())
y_pred = rf_tune.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

```
Mean Squared Error: 0.0016449261580888871
R-squared: 0.978250394312416
```

In [28]:
```
#slope dataset-1
```

In [62]:
```
x = data1['in-vitro']
y = data1['in-vivo']
slope, intercept = np.polyfit(x, y, 1)

# Print the slope
print("Slope:", slope)

# Calculate the residuals
residuals = y - (slope * x + intercept)
std_residuals = np.std(residuals)
mean_slope = np.mean(slope)
CV = (std_residuals / mean_slope) * 100

print("CV%:", CV)
```

```
Slope: 1.0036922433158615
CV%: 18.62990282217695
```

In [30]:
```
slope = 1.0036922433158615
y = [(slope * xi) for xi in x]

# Print predicted y values
print(y)
```

```
[0.15055383649737922, 0.3512922851605515, 0.4616984319252963, 0.5821415011231996, 0.8832491741179581, 0.9735814760163856, 1.0036922433158615, 0.3211815178610757, 0.44162458705897906, 0.5520307338237239, 0.9334337862837513, 1.0036922433158615, 0.31114459542791706, 0.4115138197595032, 0.5118830440910894, 0.7628061049200547, 0.25092306082896537, 0.3211815178610757, 0.37136613002686875, 0.6022153459895169, 0.3512922851605515, 0.5520307338237239, 0.6222891908558341, 0.8029537946526892]
```

In [ ]:
```
#slope for Dataset-2
```

In [34]:
```
x = data2['in-vitro']
y = data2['in-vivo']
slope, intercept = np.polyfit(x, y, 1)

# Print the slope
print("Slope:", slope)
# Calculate the residuals
residuals = y - (slope * x + intercept)
std_residuals = np.std(residuals)
mean_slope = np.mean(slope)
CV = (std_residuals / mean_slope) * 100
# Print the CV%
print("CV%:", CV)
```

```
Slope: 1.720400073632538
CV%: 3.8690352042126133
```

In [32]:
```
slope = 1.720400073632538
y = [(slope * xi) for xi in x]

# Print predicted y values
print(y)
```

```
[0.03440800147265076, 0.13763200589060304, 0.3440800147265076, 0.5677320242987376, 0.7913840338709676, 1.11826004786
11497]
```

In [ ]:
```
#update the dataset and run program again
```

In [60]:
```
data_update = pd.read_csv('D:/class/Project/ML/sensitivity analysis/updatedataset.csv')
```

In [61]:
```
import pandas as pd
import matplotlib.pyplot as plt

# create a dataframe with x, y, and coat columns
x = data_update[['in-vitro']]
y = data_update[['in-vivo']]
time1 = data_update[['time']]
data = pd.DataFrame({'x': x, 'y': y, 'time': time1}, index=[0])
#data = pd.DataFrame(data_dict)
print(data)
# group the data by coat type
grouped_data = data.groupby('time')

# plot each group separately
for time_type, group in grouped_data:
    plt.plot(group['x'], group['y'], label=time_type)

#set x and y labels
plt.xlabel('X axis')
plt.ylabel('Y axis')

# add a legend
plt.legend()

# show the graph
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [61], in <cell line: 8>()
      6 y = data_update[['in-vivo']]
      7 time1 = data_update[['time']]
----> 8 data = pd.DataFrame({'x': x, 'y': y, 'time': time1}, index=[0])
      9 #data = pd.DataFrame(data_dict)
     10 print(data)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:636, in DataFrame.__init__(self, data, index, columns, dtype, copy)
    630     mgr = self._init_mgr(
    631         data, axes={"index": index, "columns": columns}, dtype=dtype, copy=copy
    632     )
    634 elif isinstance(data, dict):
    635     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 636     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=copy, typ=manager)
    637 elif isinstance(data, ma.MaskedArray):
    638     import numpy.ma.mrecords as mrecords

File ~\anaconda3\lib\site-packages\pandas\core\internals\construction.py:502, in dict_to_mgr(data, index, columns, dtype, typ, copy)
    494     arrays = [
    495         x
    496         if not hasattr(x, "dtype") or not isinstance(x.dtype, ExtensionDtype)
    497         else x.copy()
    498         for x in arrays
    499     ]
    500     # TODO: can we get rid of the dt64tz special case above?
--> 502 return arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ, consolidate=copy)

File ~\anaconda3\lib\site-packages\pandas\core\internals\construction.py:125, in arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ, consolidate)
    122         index = ensure_index(index)
    124     # don't force copy because getting jammed in an ndarray anyway
--> 125     arrays = _homogenize(arrays, index, dtype)
    126     # _homogenize ensures
    127     #  - all(len(x) == len(index) for x in arrays)
    128     #  - all(x.ndim == 1 for x in arrays)
   (...)
    131
```

```
    132 else:
    133     index = ensure_index(index)

File ~\anaconda3\lib\site-packages\pandas\core\internals\construction.py:625, in _homogenize(data, index, dtype)
    622             val = dict(val)
    623         val = lib.fast_multiget(val, oindex._values, default=np.nan)
--> 625     val = sanitize_array(
    626         val, index, dtype=dtype, copy=False, raise_cast_failure=False
    627     )
    628     com.require_length_match(val, index)
    630 homogenized.append(val)

File ~\anaconda3\lib\site-packages\pandas\core\construction.py:598, in sanitize_array(data, index, dtype, copy, rais
e_cast_failure, allow_2d)
    595             subarr = cast(np.ndarray, subarr)
    596             subarr = maybe_infer_to_datetimelike(subarr)
--> 598 subarr = _sanitize_ndim(subarr, data, dtype, index, allow_2d=allow_2d)
    600 if isinstance(subarr, np.ndarray):
    601     # at this point we should have dtype be None or subarr.dtype == dtype
    602     dtype = cast(np.dtype, dtype)

File ~\anaconda3\lib\site-packages\pandas\core\construction.py:649, in _sanitize_ndim(result, data, dtype, index, al
low_2d)
    647     if allow_2d:
    648         return result
--> 649     raise ValueError("Data must be 1-dimensional")
    650 if is_object_dtype(dtype) and isinstance(dtype, ExtensionDtype):
    651     # i.e. PandasDtype("O")
    653     result = com.asarray_tuplesafe(data, dtype=np.dtype("object"))

ValueError: Data must be 1-dimensional
```

In [ ]: