

```

import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
from matplotlib import pyplot as plt
import seaborn as sns

# Define an exponential function to fit the data
def exponential_func(x, a, b, c):
    return a * np.exp(-b * x) + c

def process_dataset(dataset, culture_volume, mc_conc, enzyme_volume_ratio,
enzyme_conc):
    # Filter the dataset for the exponential fit
    filtered_dataset = dataset[(dataset['Time'] >= 4) & (dataset['Time']
<= 7)]

    x = filtered_dataset['Time'].values
    y = filtered_dataset['Cell density'].values

    # Fit the exponential curve to the normalized data
    p0 = (50, 0, 1) # Example initial guesses
    popt, _ = curve_fit(exponential_func, x, y, p0=p0, maxfev=10000)

    a, b, c = popt
    signal_at_20 = exponential_func(20, *popt)
    if signal_at_20 < 0:
        signal_at_20 = 0

    # Determine the detachment status
    if signal_at_20 < 10:
        status = "Fast and complete detachment"
    elif 10 <= signal_at_20 < 20:
        status = "Low activity enzyme but complete detachment"
    else:
        status = "No active enzyme, no detachment"

    enzyme_conc_in_solution = (mc_conc * enzyme_volume_ratio)*
enzyme_conc/ (culture_volume*0.2+(mc_conc * enzyme_volume_ratio))

    result = {
        "Initial signal value": a,
        "slope of signal decay": b,
        "constant coefficient": c,

```

```

        "predicted_signal_at_20": signal_at_20,
        "enzyme_conc_in_solution": enzyme_conc_in_solution,
        "status": status}

    return result

#Calculating the actual value of the dataset
def calculate_actual_signal_at_20(datasets):
    actual_values = []

    for dataset in datasets:
        # Record the actual signal value at 20 minutes
        if 20 in dataset['Time'].values:
            actual_value = dataset.loc[dataset['Time'] == 20, 'Cell
density'].values[0]
        else:
            # Find the closest time points around 20 minutes
            before = dataset[dataset['Time'] < 20].iloc[-1]
            after = dataset[dataset['Time'] > 20].iloc[0]

            # Linear interpolation
            t1, y1 = before['Time'], before['Cell density']
            t2, y2 = after['Time'], after['Cell density']
            actual_value = y1 + (20 - t1) * (y2 - y1) / (t2 - t1)

        actual_values.append(actual_value)

    return actual_values

#reading the data set for evaluating the model performance
df1=pd.read_excel("/df- MA-23045-1,65-permittivity.xlsx")
df2=pd.read_excel("/df- MA-23047-1,65-permittivity.xlsx")
df3=pd.read_excel("/df- MA-23046-1,29-permittivity.xlsx")
df4=pd.read_excel("/df- MA-23050-0,86-permittivity.xlsx")
df5=pd.read_excel("/df- MA-23049-0,82-permittivity.xlsx")
df6=pd.read_excel("/df- MA-23049-0,64-permittivity.xlsx")
df7=pd.read_excel("/df- MA-23050-0,43-permittivity.xlsx")

culture_volume=700
mc_conc=3

Label_1=["1.65 mg/mL", "1.65 mg/mL", "1.30 mg/mL", "0.86 mg/mL",
         "0.82 mg/mL", "0.64 mg/mL", "0.43 mg/mL"]

datasets=[df1, df2, df3, df4, df5, df6, df7]

```

```
Enzyme_volume_ratio=[90, 90, 50, 50, 90, 50, 50]

Initial_Enzyme_conc=[2.5, 2.5, 2.5, 1.67, 1.25, 1.25, 0.833]
```

```
# Predicted signal values and statuses
results = []
for i, dataset in enumerate(datasets):
    result = process_dataset(dataset, culture_volume, mc_conc,
Enzyme_volume_ratio[i], Initial_Enzyme_conc[i])
    results.append(result)
```

```
# outcome
```

	Initial signal value	slope of signal decay	constant coefficient	predicted signal_at_20	enzyme_conc_in_solution	status	actual signal_at_20
0	194.858744	0.296500	5.891191	6.409221	1.646341	Fast and complete detachment	4.962339
1	154.910453	0.236202	-3.347145	0.000000	1.646341	Fast and complete detachment	7.130350
2	76.417408	0.269919	3.081656	3.427362	1.293103	Fast and complete detachment	4.500000
3	180.784289	0.142606	-22.462567	0.000000	0.863793	Fast and complete detachment	6.542387
4	183.198375	0.259639	2.348334	3.366289	0.823171	Fast and complete detachment	7.624539
5	121.703973	0.171012	11.724846	15.705102	0.646552	Low activity enzyme but complete detachment	15.704630
6	60.174390	0.227782	49.713727	50.346029	0.430862	No active enzyme, no detachment	55.624667

```
# Create a DataFrame to hold actual and predicted values
```

```
results_df = pd.DataFrame(results)
results_df['actual signal_at_20'] =
calculate_actual_signal_at_20(datasets)
results_df
```

```
# Define colors based on status
```

```
status_colors = {
    "Fast and complete detachment": "green",
    "Low activity enzyme but complete detachment": "orange",
    "No active enzyme, no detachment": "red"
}
```

```
# Create scatter plot
```

```
label_an=["(1)", "(2)", "(3)", "(4)",
          "(5)", "(6)", "(7)"]
actual_status=["1.65 mg/mL", "1.65 mg/mL", "1.30 mg/mL", "0.86 mg/mL",
              "0.82 mg/mL", "0.64 mg/mL", "0.43 mg/mL"]
```

```
plt.figure(figsize=(7.5, 7.5))
```

```
for status in status_colors.keys():
    subset = results_df[results_df['status'] == status]
    plt.scatter(subset['actual signal_at_20'], subset['predicted
signal_at_20'], color=status_colors[status], label=status)
```

```

plt.plot([0, max(results_df['actual signal_at_20'])], [0,
max(results_df['actual signal_at_20'])], color='blue', linestyle='--') #
Line of equality

def annotate_points(x, y, labels, ax):
    from adjustText import adjust_text

    texts = []
    for i, txt in enumerate(labels):
        offset = (0, 2) if i % 2 == 0 else (0, -1)
        texts.append(ax.text(x[i], y[i], txt, fontsize=8, ha='center',
va='center', fontweight='bold'))
        texts[-1].set_position((x[i] + offset[0], y[i] + offset[1]))

    adjust_text(texts, only_move={'points':'y', 'text':'y'},
arrowprops=dict(arrowstyle="->", color='r', lw=0.5))

ax = plt.gca()

annotate_points(results_df['actual signal_at_20'], results_df['predicted
signal_at_20'], label_an, ax)

plt.xlabel('Actual Signal at 20 minutes', fontweight='bold')
plt.ylabel('Predicted Signal at 20 minutes', fontweight='bold')
#plt.title('Actual vs Predicted Signal at 20 minutes')
legend=plt.legend(loc="upper left", frameon=False, title="predicted
status")
plt.setp(legend.get_title(), multialignment='right')

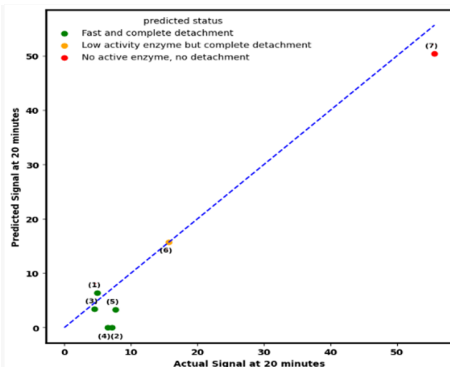
plt.xticks(fontweight='bold')
plt.yticks(fontweight='bold')

ax = plt.gca() # Get current axis
ax.spines['top'].set_linewidth(2)
ax.spines['right'].set_linewidth(2)
ax.spines['bottom'].set_linewidth(2)
ax.spines['left'].set_linewidth(2)

plt.show()

# outcome

```



```
# Calculate accuracy metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error
mae = mean_absolute_error(results_df['actual signal_at_20'],
results_df['predicted signal_at_20'])
mse = mean_squared_error(results_df['actual signal_at_20'],
results_df['predicted signal_at_20'])
rmse = np.sqrt(mse)
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")

# output
Mean Absolute Error (MAE): 3.6756594500835553
Mean Squared Error (MSE): 20.412207375534198
Root Mean Squared Error (RMSE): 4.517987093334177

#make a function for comparing the actual and prediction signal value
def prediction_actual_table (df, culture_volume, MC_conc,
Enzyme_volume_ratio, Initial_Enzyme_con ):
    result_df=pd.DataFrame([process_dataset(df, culture_volume, MC_conc,
Enzyme_volume_ratio, Initial_Enzyme_con)])
    result_df['actual signal_at_20']= calculate_actual_signal_at_20([df])
    return result_df

#Evaluating the model performance on new data set
df8=pd.read_excel("/df- MA-23047-1,29 - low activity-permitivit.xlsx")

result_df8=pd.DataFrame([process_dataset(df8, 700, 3, 50, 2.5)])

from IPython.display import display, HTML
# Display the DataFrame with centered column names
def display_centered_df(df):
    style = df.style.set_table_styles(
```

```

        [{'selector': 'th', 'props': [('text-align', 'center')]}]
    ).set_properties(**{'text-align': 'center'})
    display(style)

```

```
display_centered_df(result_df8)
```

```
# output
```

	Initial signal value	slope of signal decay	constant coefficient	predicted signal_at_20	enzyme_conc_in_solution	status
0	126.977332	0.124130	7.281356	17.887188	1.293103	Low activity enzyme but complete detachment

```
prediction_actual_table (df8, 700, 3, 50, 2.5)
```

```
# output
```

	Initial signal value	slope of signal decay	constant coefficient	predicted signal_at_20	enzyme_conc_in_solution	status	actual signal_at_20
0	126.977332	0.12413	7.281356	17.887188	1.293103	Low activity enzyme but complete detachment	17.221419

```
#plotting the predicted graph vs actual value for the one with low activity
```

```
result=process_dataset(df8, 700, 3, 50, 2.5)
```

```

a, b, c , predicted_status, predicted_signal_at20= result['Initial signal value'], result['slope of signal decay'], result['constant coefficient'], result['status'], result['predicted signal_at_20']

```

```
predicted_signal=[]
```

```
Time=[]
```

```
for time in range(0,60,1):
```

```
    predicted_signal.append(exponential_func(time, a, b, c))
```

```
    Time.append(time)
```

```
plt.plot(Time, predicted_signal, color='blue', label='Predicted Signal')
```

```
plt.plot (df8['Time'], df8['Cell density'], color='red', label='Actual Signal')
```

```
plt.xlim(0, 60)
```

```
plt.ylim(0, 140)
```

```
plt.xlabel('Detachment course time (min)', fontweight='bold')
```

```
plt.ylabel('Permittivity signal (pF/cm)', fontweight='bold')
```

```
plt.xticks(fontweight='bold')
```

```
plt.yticks(fontweight='bold')
```

```
plt.legend(bbox_to_anchor=(1, 0.94))
```

```

plt.text(4, 133, f'Predicted Status: {predicted_status}', fontsize=10,
bbox=dict(facecolor='white', alpha=0.5))

```

```
# Customize plot border
```

```
ax = plt.gca() # Get current axis
```

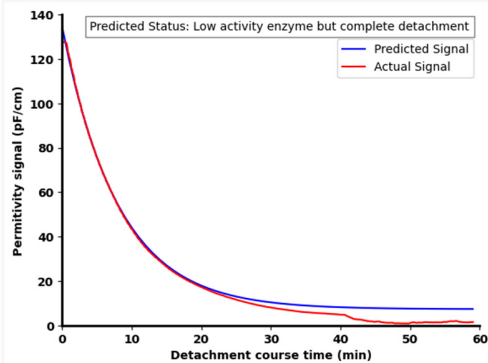
```
ax.spines['top'].set_visible(False)
```

```
ax.spines['right'].set_visible(False)
```

```
ax.spines['bottom'].set_linewidth(2)
ax.spines['left'].set_linewidth(2)
```

```
plt.show()
```

```
# output
```



```
#Reading the dataset from the new generation sensor
```

```
df9=pd.read_excel("/df- MA-23050-0,86 - new probe-permittivity.xlsx")
```

```
df10=pd.read_excel("/df- MA-23050-0,43 - new sensor-permittivity.xlsx")
```

```
#Prediction result for the new sensor
```

```
prediction_actual_table (df9, 700, 3, 50, 1.67)
```

```
{'slope of signal decay': 0.1677032388293067,
 'predicted signal_at_20': 0,
 'enzyme_conc_in_solution': 0.8637931034482759,
 'status': 'Fast and complete detachment'}
```

```
prediction_actual_table (df10, 700, 3, 50, 0.833)
```

```
{'slope of signal decay': -5.603199349111244e-05,
 'predicted signal_at_20': 57.112781383182664,
 'enzyme_conc_in_solution': 0.4308620689655172,
 'status': 'No active enzyme, no detachment'}
```