In [18]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [19]:
```
df = pd.read_csv('iris.csv')
```

In [20]:
```
print("-----------Dataframe Info------------")
print(df.info())
print("\n")
```

```
-----------Dataframe Info------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

In [21]:
```
print("-----------Dataframe Describe------------")
print(df.describe())
print("\n")
```

```
-----------Dataframe Describe------------
               Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthC
m
count  150.000000     150.000000    150.000000     150.000000    150.00000
0
mean    75.500000       5.843333      3.054000       3.758667      1.19866
7
std     43.445368       0.828066      0.433594       1.764420      0.76316
1
min      1.000000       4.300000      2.000000       1.000000      0.10000
0
25%     38.250000       5.100000      2.800000       1.600000      0.30000
0
50%     75.500000       5.800000      3.000000       4.350000      1.30000
0
75%    112.750000       6.400000      3.300000       5.100000      1.80000
0
max    150.000000       7.900000      4.400000       6.900000      2.50000
0
```

In [22]:
```
print("----------Dataframe Head-----------")
print(df.head())
print("\n")
```

```
----------Dataframe Head-----------
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm Species
0   1            5.1           3.5            1.4           0.2  setosa
1   2            4.9           3.0            1.4           0.2  setosa
2   3            4.7           3.2            1.3           0.2  setosa
3   4            4.6           3.1            1.5           0.2  setosa
4   5            5.0           3.6            1.4           0.2  setosa
```

In [23]:
```
print("---------------Data Preprocessing----------------")
X = df.iloc[:,0:4]
Y = df['Species'].values
```

```
---------------Data Preprocessing---------------
```

In [24]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, ra
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [26]:
```
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test  Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')
print("\n")
```

```
Train Dataset Size - X: (120, 4), Y: (120,)
Test  Dataset Size - X: (30, 4), Y: (30,)
```

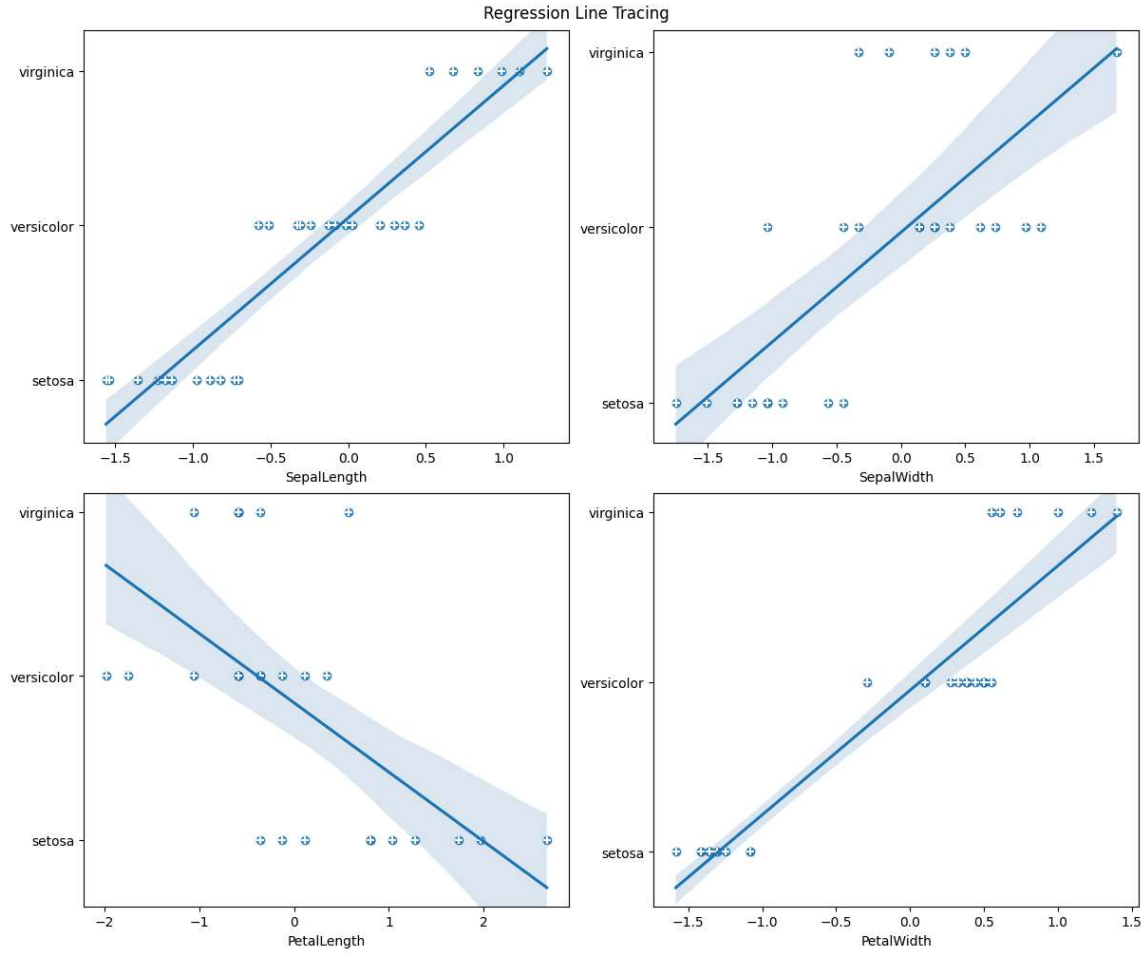
In [27]:
```
print("--------------Naive Bayes Classifier----------------------")
# This code fits a Naive Bayes classifier model on the training data, makes
# maps the predicted species labels to integers, and plots regression lines
# to the actual labels for each of the 4 feature columns. It shows how the
# predicting the species from each individual feature.
from sklearn.naive_bayes import GaussianNB
```

```
--------------Naive Bayes Classifier----------------------
```

In [28]:
```
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)
```

In [29]:
```
mapper = {'setosa': 0, 'versicolor': 1, 'virginica': 2}
predictions_ = [mapper[i] for i in predictions]
```

In [30]:
```
fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True
_ = fig.suptitle('Regression Line Tracing')
for i in range(4):
    x, y = i // 2, i % 2
    _ = sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
    _ = axs[x, y].scatter(X_test[:, i][::-1], Y_test[::-1], marker = '+', c
    _ = axs[x, y].set_xlabel(df.columns[i + 1][:-2])
plt.show()
print("\n")
```



In [ ]:
```
```

In [31]:
```
print("------------Confusion Matrix-------------")
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
------------Confusion Matrix-------------
```

In [32]:
```
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
               | Positive Prediction\t| Negative Prediction
---------------+-----------------------+---------------------
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0
---------------+-----------------------+---------------------
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1
```

```
Confusion matrix :

               | Positive Prediction    | Negative Prediction
---------------+------------------------+---------------------
Positive Class | True Positive (TP) 11  | False Negative (FN) 0
---------------+------------------------+---------------------
Negative Class | False Positive (FP) 0  | True Negative (TN) 13
```

In [33]:
```
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

```
Classification report :
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```