```
# DAA Project – Transmission Line Fault Detection and Classification
**BT23CSE001, BT23CSE016, BT23CSE038**

## Introduction

Electrical power transmission systems are prone to various types of
faults such as single line-to-ground (LG), line-to-line (LL), double
line-to-ground (LLG), and three-phase (LLL) faults. These faults, if
not detected and cleared promptly, can cause severe damage to
equipment, blackouts, and safety hazards.

In this project, we focus on the detection and classification of
transmission line faults using Artificial Neural Networks (ANN). The
use of ANN models provides a robust and efficient approach for
analyzing fault data and accurately identifying fault types based on
input parameters like voltage and current waveforms.

By leveraging machine learning techniques, particularly supervised
learning via ANN architectures, the model can learn to distinguish
between different fault types, enhancing the reliability and
automation of fault analysis in power systems.
```

Electrical Fault detection and Classification using ANN models

```python
# Importing necessary packages
import pandas as pd
import numpy as np
import sklearn
from sklearn import linear_model
import matplotlib.pyplot as plt

from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score,f1_score
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline

from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC,SVC
from sklearn.neighbors import KNeighborsClassifier

#Importing the data
detection_train = pd.read_excel('detect_dataset.xlsx').dropna(axis=1)
```

```
class_train = pd.read_csv('classData.csv').dropna(axis=1)

features=['Ia','Ib','Ic','Va','Vb','Vc']
class_target = ['G','C','B','A']

#Defining the inputs and outputs
detection_data_X = detection_train[features]
class_data_X = class_train[features]
detection_data_Y = detection_train['Output (S)']
class_data_Y = class_train[class_target]

#Defining accuracy and error vectors
detect_accuracy = list()
detect_error = list()
class_accuracy = list()
class_error = list()

#Splitting the data
class_train_X,class_test_X,class_train_Y,class_test_Y=
train_test_split(class_data_X,class_data_Y,test_size=0.33,random_state
=1)
detection_train_X,detection_test_X,detection_train_Y,detection_test_Y
=
train_test_split(detection_data_X,detection_data_Y,test_size=0.33,rand
om_state=1)
```

Linear regression

```
#Defining different Models for different classification problems
detection_model = linear_model.Lasso(alpha = 2.0)
class_model = LinearRegression()

#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y =
np.array([class_train_Y['G']*1+class_train_Y['A']*2+class_train_Y['B']
*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
LinearRegression()
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
print('The Error of our Detection Model is:
',mean_squared_error(detection_test_Y,detection_preds))
```

```
class_Y = np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+class_test_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is: ',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_preds))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.24375743622444437
The Error of our Classification Model is:  17.301569015218817
```

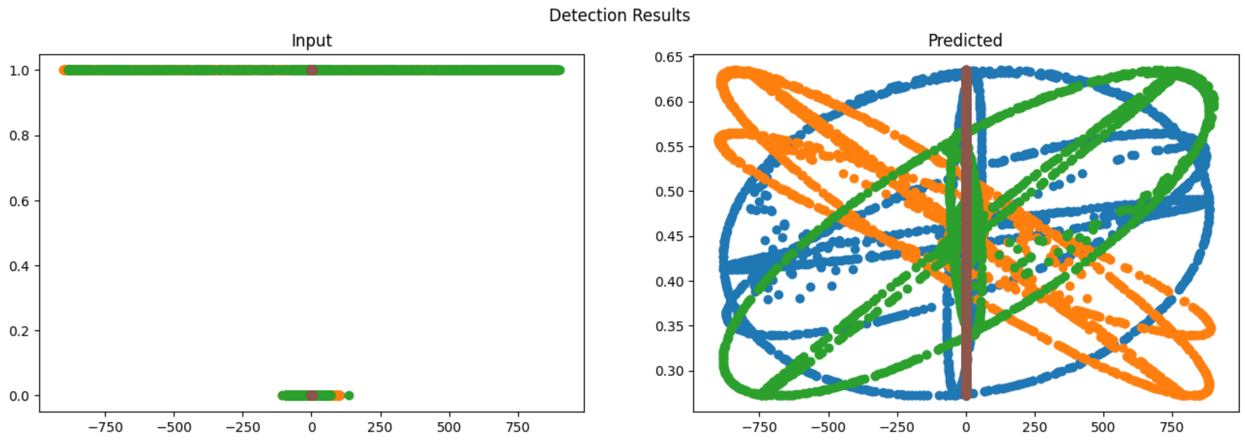
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',(detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ',(class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detection_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.017945755271112085
The accuracy score of our Classification Model is:  0.03349707430965532
```

Graphs

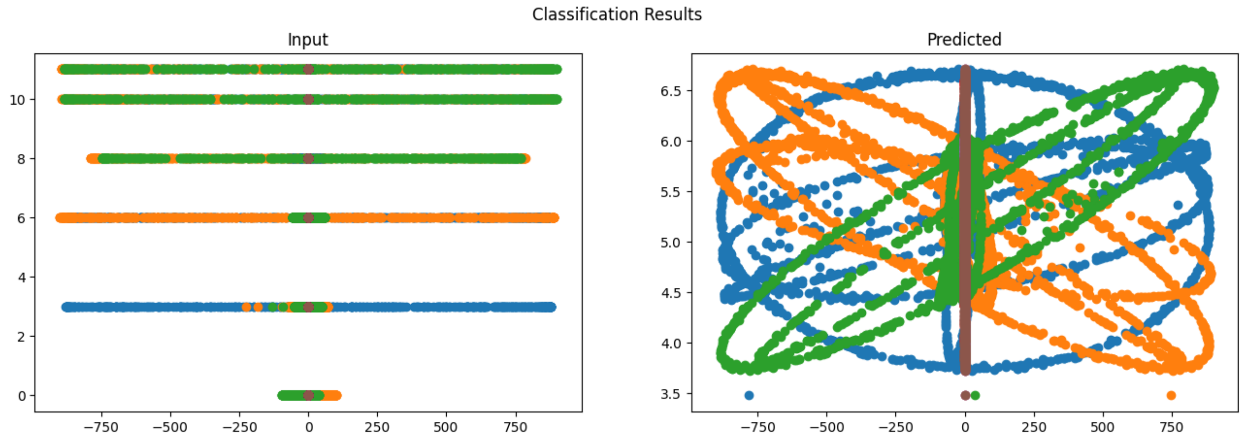
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f407847d0>,
 <matplotlib.lines.Line2D at 0x7f0f40784910>,
 <matplotlib.lines.Line2D at 0x7f0f40784a50>,
 <matplotlib.lines.Line2D at 0x7f0f40784b90>,
 <matplotlib.lines.Line2D at 0x7f0f40784cd0>,
 <matplotlib.lines.Line2D at 0x7f0f40784e10>]
```



```python
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f408cc7d0>,
 <matplotlib.lines.Line2D at 0x7f0f408cc910>,
 <matplotlib.lines.Line2D at 0x7f0f408cca50>,
 <matplotlib.lines.Line2D at 0x7f0f408ccb90>,
 <matplotlib.lines.Line2D at 0x7f0f408cccd0>,
 <matplotlib.lines.Line2D at 0x7f0f408cce10>]
```



Logistic regression

```python
#Defining different Models for different classification problems
detection_model = LogisticRegression(max_iter=5000)
class_model = LogisticRegression(max_iter=5000)
```

```
#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y = np.array([class_train_Y['G']*1+class_train_Y['A']
                    *2+class_train_Y['B']*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
LogisticRegression(max_iter=5000)
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
print('The Error of our Detection Model is:
',mean_squared_error(detection_test_Y,detection_preds))

class_Y =
np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+
class_test_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is:
',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_pred
s))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.26155011360767483
The Error of our Classification Model is:  42.65895953757225
```

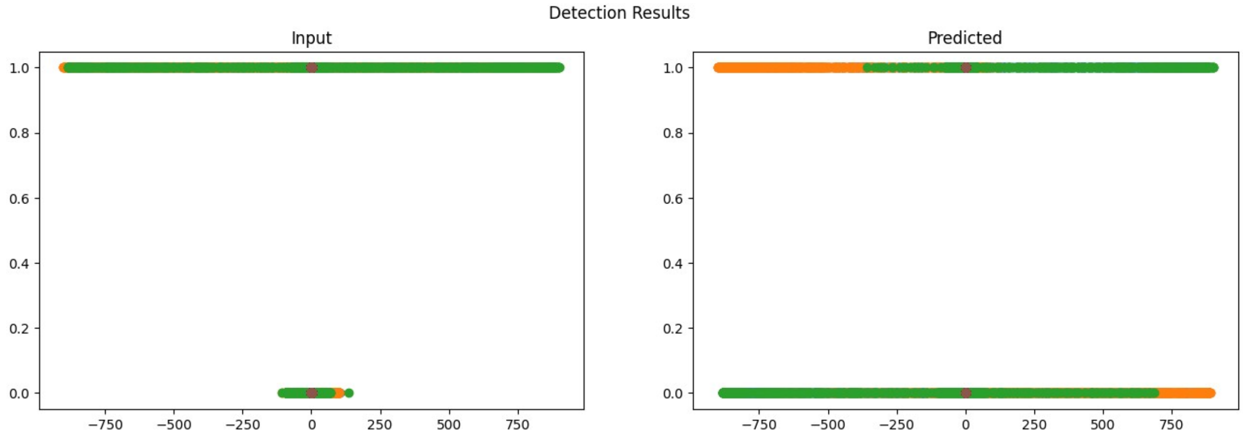
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',
(detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ',
(class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detecti
on_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.7384498863923251
The accuracy score of our Classification Model is:
0.32524084778420037
```

Graphs

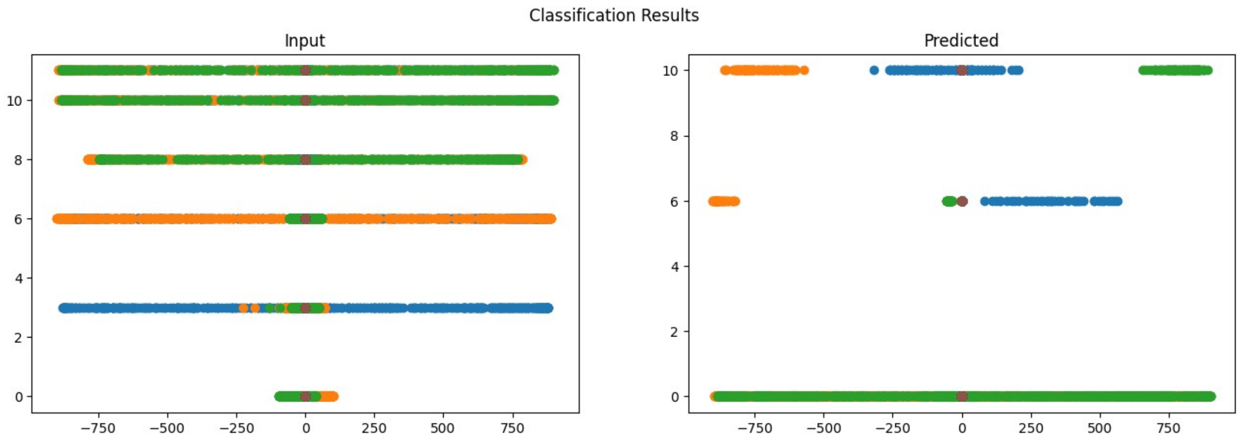
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f40b5fb10>,
 <matplotlib.lines.Line2D at 0x7f0f40b8ac10>,
 <matplotlib.lines.Line2D at 0x7f0f40b8ad50>,
 <matplotlib.lines.Line2D at 0x7f0f40b8ae90>,
 <matplotlib.lines.Line2D at 0x7f0f40b8afd0>,
 <matplotlib.lines.Line2D at 0x7f0f40b8b110>]
```



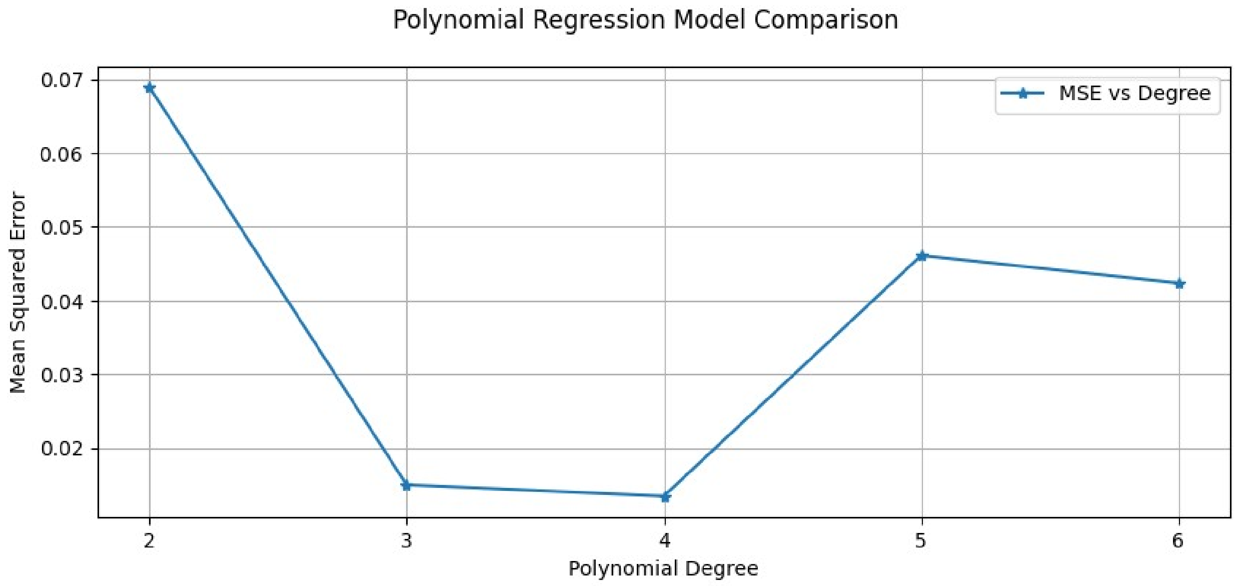
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f40b0c190>,
 <matplotlib.lines.Line2D at 0x7f0f40b0c2d0>,
 <matplotlib.lines.Line2D at 0x7f0f40b0c410>,
 <matplotlib.lines.Line2D at 0x7f0f40b0c550>,
 <matplotlib.lines.Line2D at 0x7f0f40b0c690>,
 <matplotlib.lines.Line2D at 0x7f0f40b0c7d0>]
```



Polynomial regression

```
# Selection of suitable polynomial degree
errors = []
degrees = list(range(2, 7))  # Degrees 2 to 6

for i in degrees:
    poly = PolynomialFeatures(i)
    model = LinearRegression()
    model.fit(poly.fit_transform(class_train_X), class_train_Y)
    preds = model.predict(poly.fit_transform(class_test_X))
    errors.append(mean_squared_error(class_test_Y, preds))

# Plotting
fig, ax = plt.subplots(1, 1, figsize=(10, 4))
fig.suptitle('Polynomial Regression Model Comparison')

# Set ticks and labels to match degree indices
ax.set_xticks(range(len(degrees)))
ax.set_xticklabels([str(d) for d in degrees])

ax.plot(errors, '*-', label='MSE vs Degree')
ax.set_ylabel('Mean Squared Error')
ax.set_xlabel('Polynomial Degree')
ax.grid(True)
ax.legend()
plt.show()
```



```python
#Defining different Models for different classification problems
detection_model = PolynomialFeatures(2)
class_model = PolynomialFeatures(4)
detect_linear = LinearRegression()
class_linear = LinearRegression()

#Fitting the data in different models
detect_linear.fit(detection_model.fit_transform(detection_train_X),detection_train_Y)
class_linear.fit(class_model.fit_transform(class_train_X),class_train_Y)
```

```
LinearRegression()
```

Results

```python
#Predicting test values and printing out Mean Squared Error
detection_preds = detect_linear.predict(detection_model.fit_transform(detection_test_X))
print('The Error of our Detection Model is: ',mean_squared_error(detection_test_Y,detection_preds))

class_preds = class_linear.predict(class_model.fit_transform(class_test_X))
print('The Error of our Classification Model is: ',mean_squared_error(class_test_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_preds))
class_error.append(mean_squared_error(class_test_Y,class_preds))
```

```
The Error of our Detection Model is:  0.03445426707454392
The Error of our Classification Model is:  0.013437887105488064
```

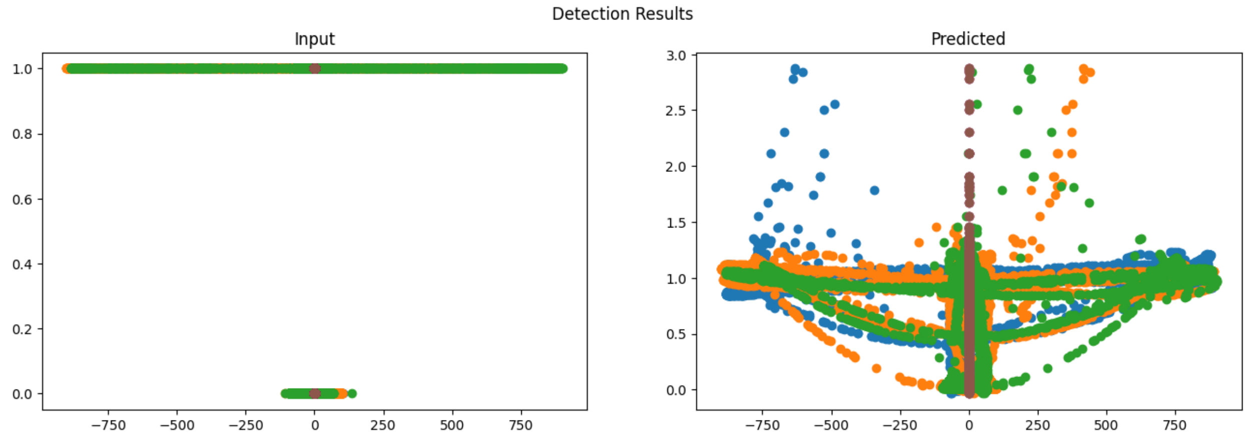
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',
(detect_linear.score(detection_model.fit_transform(detection_test_X),d
etection_test_Y)))
print('The accuracy score of our Classification Model is: ',
(class_linear.score(class_model.fit_transform(class_test_X),class_test
_Y)))

#Storing accuracy values
detect_accuracy.append((detect_linear.score(detection_model.fit_transf
orm(detection_test_X),detection_test_Y)))
class_accuracy.append((class_linear.score(class_model.fit_transform(cl
ass_test_X),class_test_Y)))
```

```
The accuracy score of our Detection Model is:  0.8611900430458109
The accuracy score of our Classification Model is:  0.9451278131164605
```

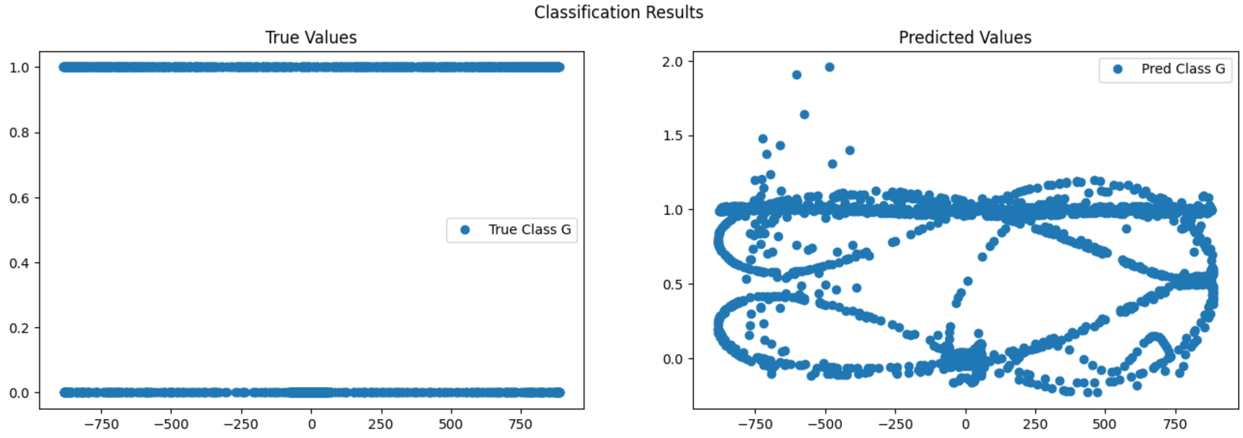
Graphs

```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f40a03d90>,
 <matplotlib.lines.Line2D at 0x7f0f409cac10>,
 <matplotlib.lines.Line2D at 0x7f0f409cad50>,
 <matplotlib.lines.Line2D at 0x7f0f409cae90>,
 <matplotlib.lines.Line2D at 0x7f0f409cafd0>,
 <matplotlib.lines.Line2D at 0x7f0f409cb110>]
```



```
fig, axs = plt.subplots(1, 2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')

axs[0].set_title('True Values')
axs[1].set_title('Predicted Values')

# Plot feature 0 vs class 0 (example)
axs[0].plot(class_test_X.iloc[:, 0], class_test_Y.iloc[:, 0], 'o',
label='True Class G')
axs[1].plot(class_test_X.iloc[:, 0], class_preds[:, 0], 'o',
label='Pred Class G')

axs[0].legend()
axs[1].legend()
plt.show()
```



Naive Bayes

```
#Defining different Models for different classification problems
detection_model = GaussianNB()
class_model = GaussianNB()

#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y = np.array([class_train_Y['G']*1+class_train_Y['A']
                    *2+class_train_Y['B']*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
GaussianNB()
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
print('The Error of our Detection Model is:
',mean_squared_error(detection_test_Y,detection_preds))

class_Y =
np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+
class_test_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is:
',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_pred
s))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.019439535470840697
The Error of our Classification Model is:  2.1078998073217727
```

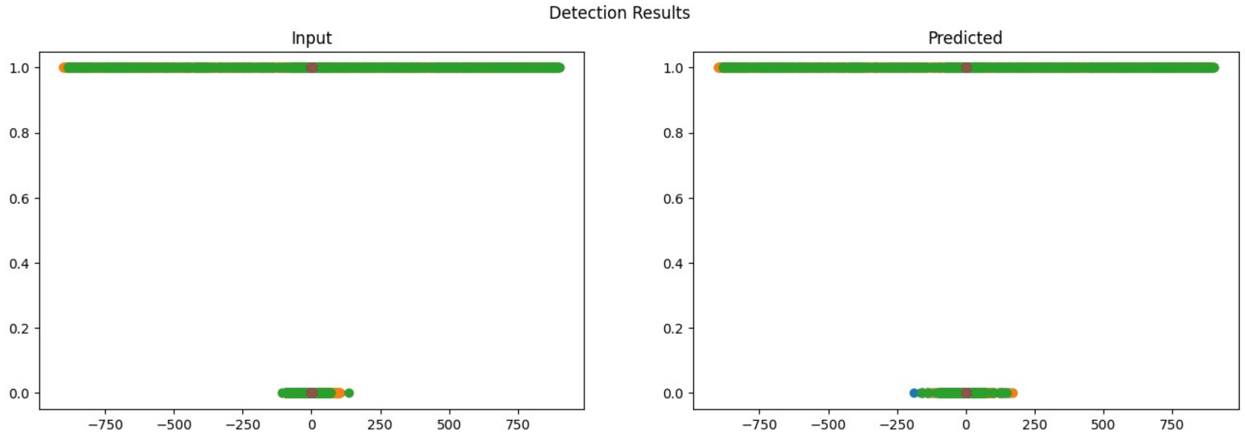
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',
(detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ',
(class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detecti
on_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.9805604645291593
The accuracy score of our Classification Model is:  0.796917148362235
```

Graphs

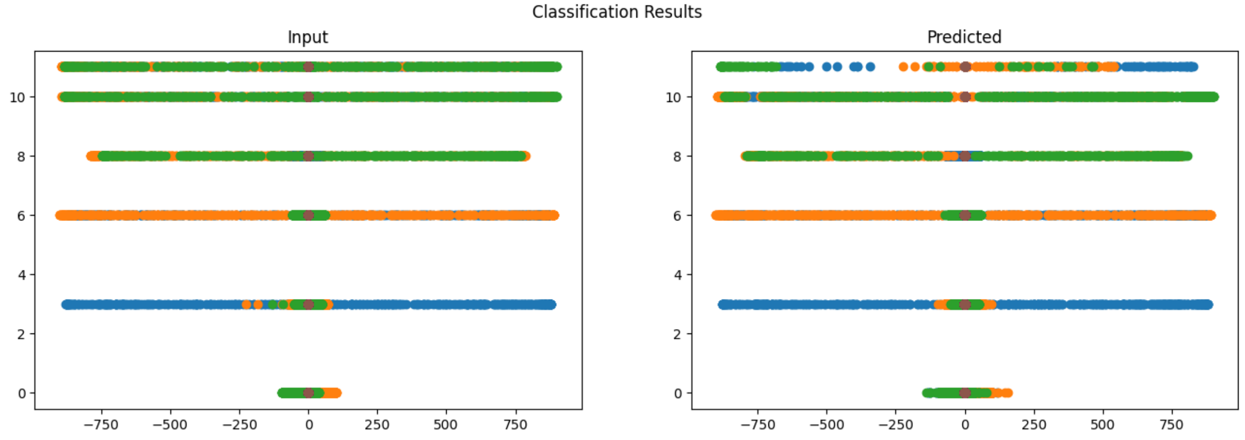
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3c5a6d50>,
 <matplotlib.lines.Line2D at 0x7f0f3c5a6e90>,
 <matplotlib.lines.Line2D at 0x7f0f3c5a6fd0>,
 <matplotlib.lines.Line2D at 0x7f0f3c5a7110>,
 <matplotlib.lines.Line2D at 0x7f0f3c5a7250>,
 <matplotlib.lines.Line2D at 0x7f0f3c5a7390>]
```



```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3c5020d0>,
 <matplotlib.lines.Line2D at 0x7f0f3c502210>,
 <matplotlib.lines.Line2D at 0x7f0f3c502350>,
 <matplotlib.lines.Line2D at 0x7f0f3c502490>,
 <matplotlib.lines.Line2D at 0x7f0f3c5025d0>,
 <matplotlib.lines.Line2D at 0x7f0f3c502710>]
```



Decision Tree classifier

```
#Defining different Models for different classification problems
detection_model = DecisionTreeClassifier()
class_model = DecisionTreeClassifier()


#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y = np.array([class_train_Y['G']*1+class_train_Y['A']
                    *2+class_train_Y['B']*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
DecisionTreeClassifier()
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
print('The Error of our Detection Model is:
',mean_squared_error(detection_test_Y,detection_preds))

class_Y =
np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+
class_test_Y['C']*5])
class_Y = class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is:
',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_pred
s))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.00555415299166877
The Error of our Classification Model is:  0.3040462427745665
```

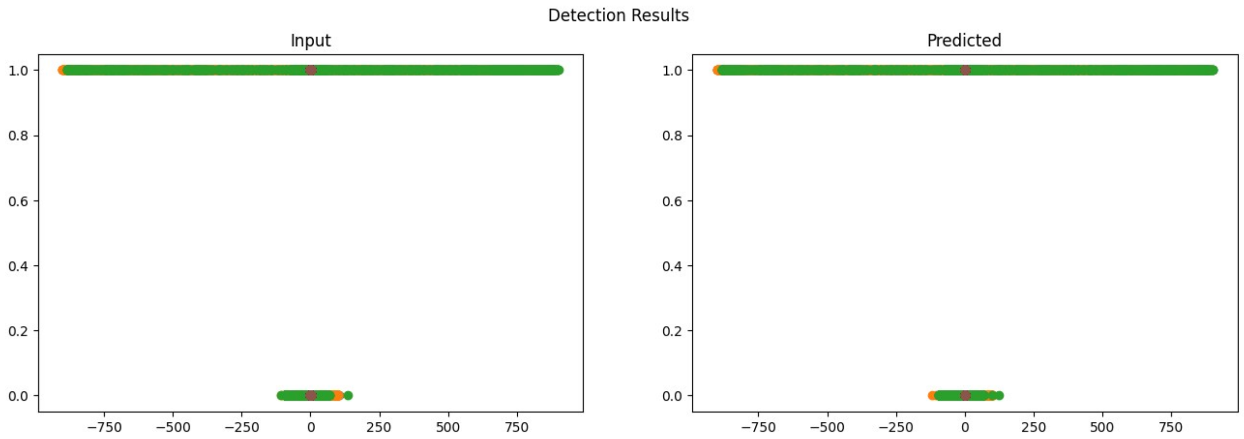
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',
(detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ',
(class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detecti
on_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.9944458470083313
The accuracy score of our Classification Model is:  0.8635838150289017
```

Graphs

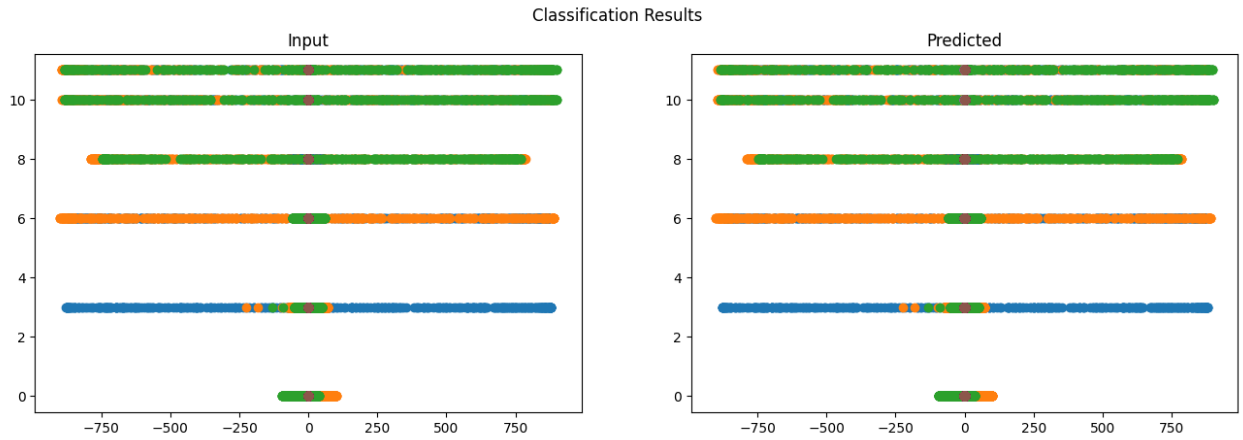
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3a938f50>,
 <matplotlib.lines.Line2D at 0x7f0f3a96fd90>,
 <matplotlib.lines.Line2D at 0x7f0f3a96fed0>,
 <matplotlib.lines.Line2D at 0x7f0f3a7b4050>,
 <matplotlib.lines.Line2D at 0x7f0f3a7b4190>,
 <matplotlib.lines.Line2D at 0x7f0f3a7b42d0>]
```



```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3a699310>,
 <matplotlib.lines.Line2D at 0x7f0f3a699450>,
 <matplotlib.lines.Line2D at 0x7f0f3a699590>,
 <matplotlib.lines.Line2D at 0x7f0f3a6996d0>,
 <matplotlib.lines.Line2D at 0x7f0f3a699810>,
 <matplotlib.lines.Line2D at 0x7f0f3a699950>]
```



SVM

```
#Defining different Models for different classification problems
detection_model = SVC()
class_model = LinearSVC()

#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y = np.array([class_train_Y['G']*1+class_train_Y['A']
                    *2+class_train_Y['B']*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
LinearSVC()
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
print('The Error of our Detection Model is:
',mean_squared_error(detection_test_Y,detection_preds))

class_Y =
np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+
class_test_Y['C']*5])
class_Y = class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is:
',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_pred
s))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.01792476647311285
The Error of our Classification Model is:  41.223121387283236
```

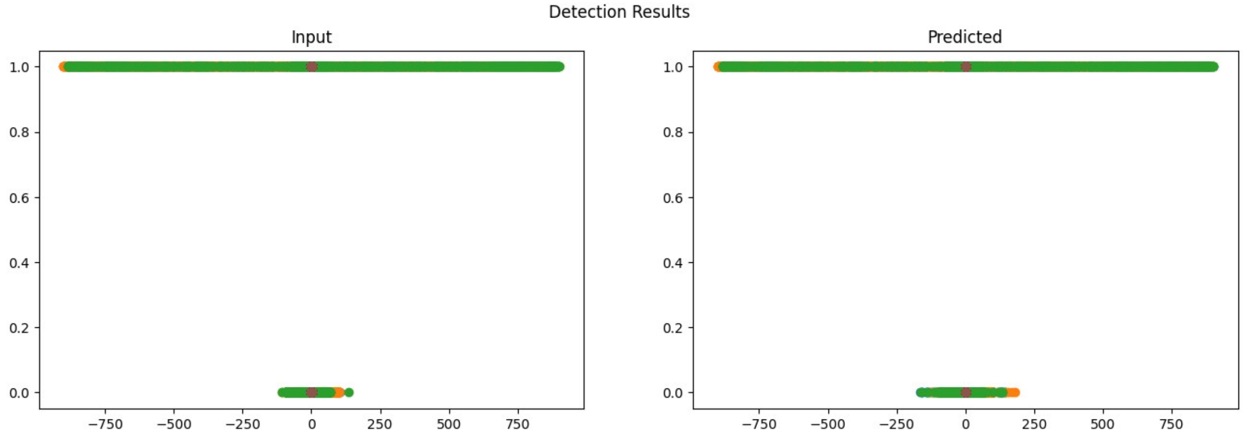
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ',
(detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ',
(class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detection_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.9820752335268872
The accuracy score of our Classification Model is:
0.31676300578034683
```

Graphs

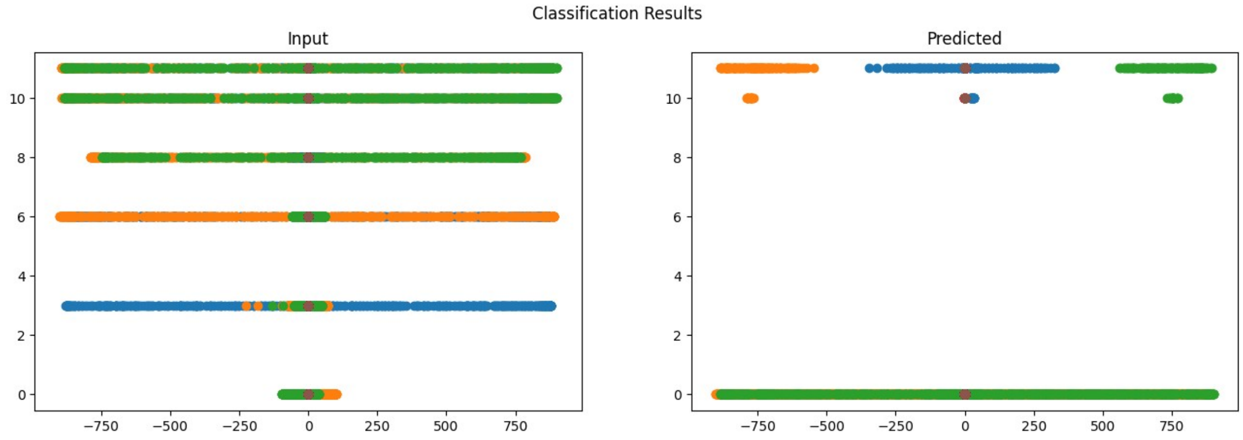
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3a938e10>,
 <matplotlib.lines.Line2D at 0x7f0f3a7720d0>,
 <matplotlib.lines.Line2D at 0x7f0f3a772210>,
 <matplotlib.lines.Line2D at 0x7f0f3a772350>,
 <matplotlib.lines.Line2D at 0x7f0f3a772490>,
 <matplotlib.lines.Line2D at 0x7f0f3a7725d0>]
```



```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3a66f610>,
 <matplotlib.lines.Line2D at 0x7f0f3a66f750>,
 <matplotlib.lines.Line2D at 0x7f0f3a66f890>,
 <matplotlib.lines.Line2D at 0x7f0f3a66f9d0>,
 <matplotlib.lines.Line2D at 0x7f0f3a66fb10>,
 <matplotlib.lines.Line2D at 0x7f0f3a66fc50>]
```



KNN

```
#Defining different Models for different classification problems
detection_model = KNeighborsClassifier(n_neighbors=2)
class_model = KNeighborsClassifier(n_neighbors=6)

#Fitting the data in different models
detection_model.fit(detection_train_X,detection_train_Y)
class_Y = np.array([class_train_Y['G']*1+class_train_Y['A']
                    *2+class_train_Y['B']*3+class_train_Y['C']*5])
class_Y= class_Y.transpose().ravel()
class_model.fit(class_train_X,class_Y)
```

```
KNeighborsClassifier(n_neighbors=6)
```

Results

```
#Predicting test values and printing out Mean Squared Error
detection_preds = detection_model.predict(detection_test_X)
```

```
print('The Error of our Detection Model is: ',mean_squared_error(detection_test_Y,detection_preds))

class_Y = np.array([class_test_Y['G']*1+class_test_Y['A']*2+class_test_Y['B']*3+class_test_Y['C']*5])
class_Y = class_Y.transpose().ravel()
class_preds = class_model.predict(class_test_X)
print('The Error of our Classification Model is: ',mean_squared_error(class_Y,class_preds))

#storing error values
detect_error.append(mean_squared_error(detection_test_Y,detection_preds))
class_error.append(mean_squared_error(class_Y,class_preds))
```

```
The Error of our Detection Model is:  0.007573844988639233
The Error of our Classification Model is:  0.9845857418111753
```

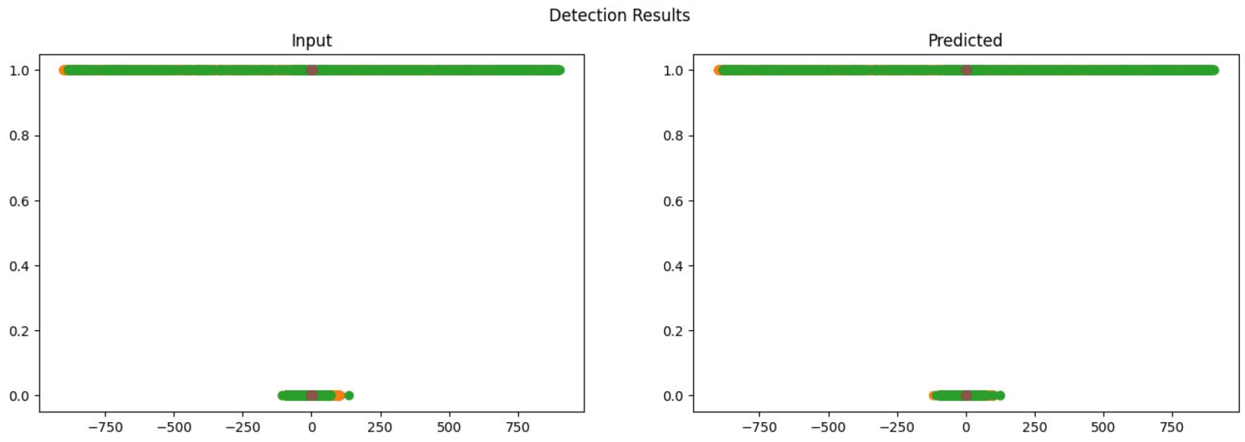
```
# Printing out accuracy scores of our models
print('The accuracy score of our Detection Model is: ', (detection_model.score(detection_test_X,detection_test_Y)))
print('The accuracy score of our Classification Model is: ', (class_model.score(class_test_X,class_Y)))

#Storing accuracy values
detect_accuracy.append((detection_model.score(detection_test_X,detection_test_Y)))
class_accuracy.append((class_model.score(class_test_X,class_Y)))
```

```
The accuracy score of our Detection Model is:  0.9924261550113608
The accuracy score of our Classification Model is:  0.8246628131021194
```

Graphs

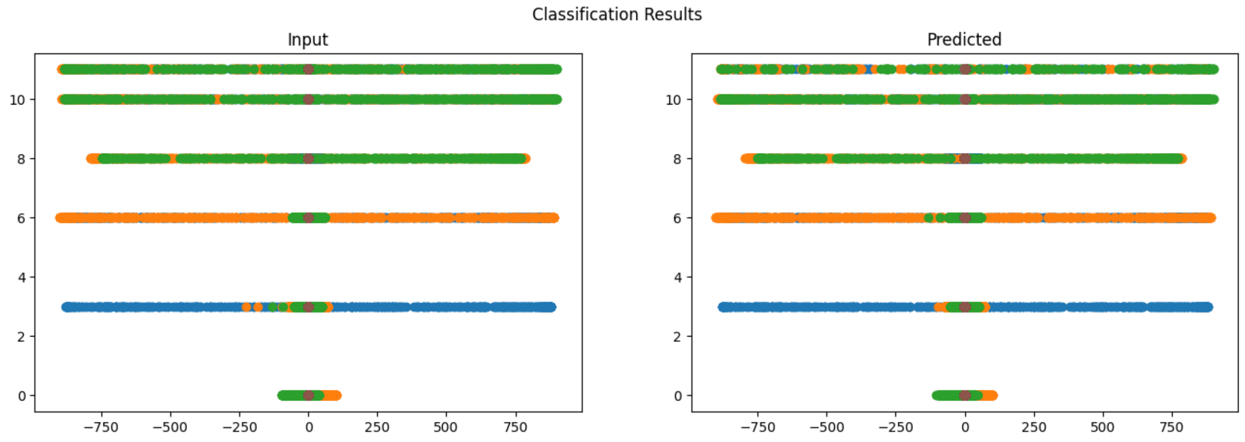
```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Detection Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(detection_test_X, detection_test_Y,'o')
axs[1].plot(detection_test_X, detection_preds,'o')
```

```
[<matplotlib.lines.Line2D at 0x7f0f3a516990>,
 <matplotlib.lines.Line2D at 0x7f0f3a38cf50>,
 <matplotlib.lines.Line2D at 0x7f0f3a38d090>,
 <matplotlib.lines.Line2D at 0x7f0f3a38d1d0>,
 <matplotlib.lines.Line2D at 0x7f0f3a38d310>,
 <matplotlib.lines.Line2D at 0x7f0f3a38d450>]
```



```
fig, axs = plt.subplots(1,2)
fig.set_figwidth(16)
fig.suptitle('Classification Results')
axs[0].set_title('Input')
axs[1].set_title('Predicted')
axs[0].plot(class_test_X, class_Y,'o')
axs[1].plot(class_test_X, class_preds,'o')
```

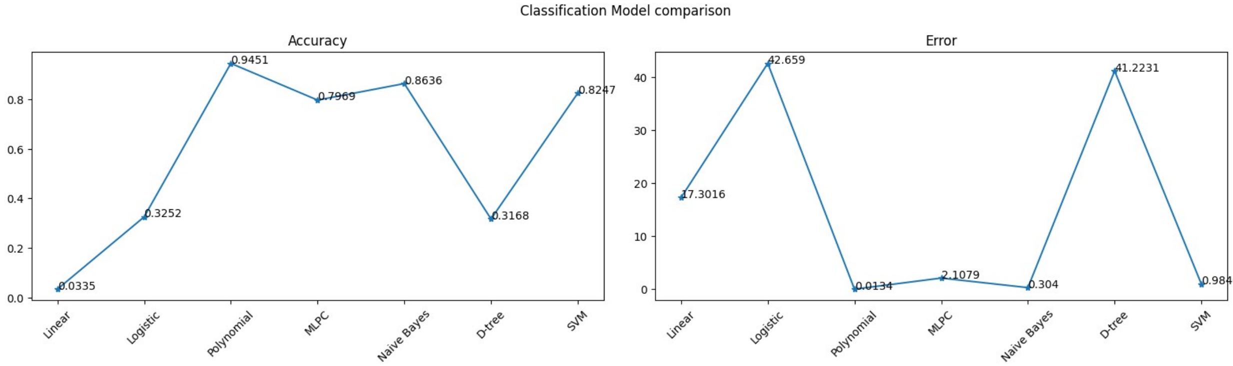
```
[<matplotlib.lines.Line2D at 0x7f0f3a472490>,
 <matplotlib.lines.Line2D at 0x7f0f3a4725d0>,
 <matplotlib.lines.Line2D at 0x7f0f3a472710>,
 <matplotlib.lines.Line2D at 0x7f0f3a472850>,
 <matplotlib.lines.Line2D at 0x7f0f3a472990>,
 <matplotlib.lines.Line2D at 0x7f0f3a472ad0>]
```



Model selection

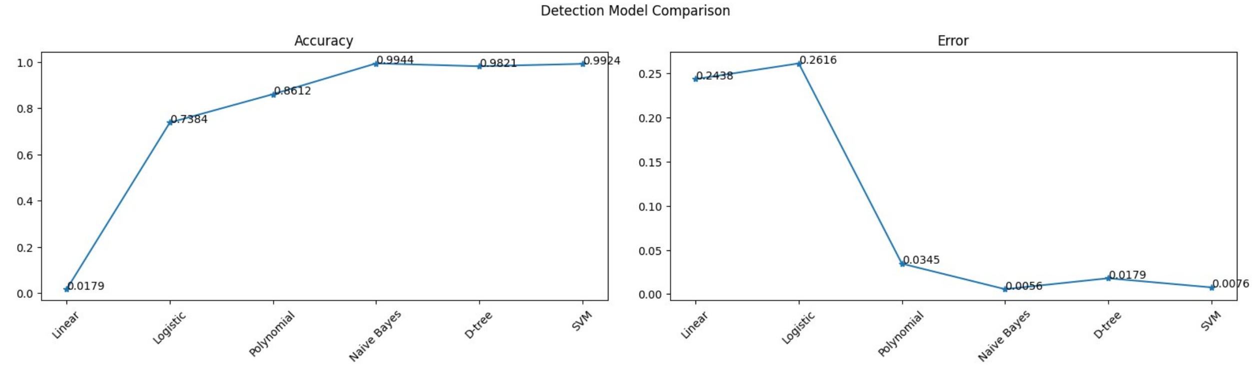
```
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(16)
fig.suptitle('Classification Model comparison')
```

```python
x = list(range(8))

model_names = ['Linear', 'Logistic', 'Polynomial', 'MLPC', 'Naive
Bayes', 'D-tree', 'SVM', 'KNN']

# Plot Accuracy
ax[0].set_xticks(x)
ax[0].set_xticklabels(model_names, rotation=45)
ax[0].set_title('Accuracy')
ax[0].plot(class_accuracy, '*-')
rounded_acc = [round(val, 4) for val in class_accuracy]
for i, j in zip(x, rounded_acc):
    ax[0].annotate(str(j), xy=(i, class_accuracy[i]))

# Plot Error
ax[1].set_xticks(x)
ax[1].set_xticklabels(model_names, rotation=45)
ax[1].set_title('Error')
ax[1].plot(class_error, '*-')
rounded_err = [round(val, 4) for val in class_error]
for i, j in zip(x, rounded_err):
    ax[1].annotate(str(j), xy=(i, class_error[i]))

plt.tight_layout()
plt.show()
```



```python
fig, ax = plt.subplots(1, 2)
fig.set_figwidth(16)
fig.suptitle('Detection Model Comparison')

# Updated labels and x-axis
model_names = ['Linear', 'Logistic', 'Polynomial', 'Naive Bayes', 'D-
tree', 'SVM', 'KNN']
x = list(range(len(model_names)))

# Remove MLPC (index 3) from detect_accuracy and detect_error
detect_accuracy_filtered = detect_accuracy[:3] + detect_accuracy[4:]
detect_error_filtered = detect_error[:3] + detect_error[4:]
```

```
# Accuracy Plot
ax[0].set_xticks(x)
ax[0].set_xticklabels(model_names, rotation=45)
ax[0].set_title('Accuracy')
ax[0].plot(detect_accuracy_filtered, '*-')
rounded_acc = [round(val, 4) for val in detect_accuracy_filtered]
for i, j in zip(x, rounded_acc):
    ax[0].annotate(str(j), xy=(i, j))

# Error Plot
ax[1].set_xticks(x)
ax[1].set_xticklabels(model_names, rotation=45)
ax[1].set_title('Error')
ax[1].plot(detect_error_filtered, '*-')
rounded_err = [round(val, 4) for val in detect_error_filtered]
for i, j in zip(x, rounded_err):
    ax[1].annotate(str(j), xy=(i, j))

plt.tight_layout()
plt.show()
```


Detection Model Comparison
Accuracy
Linear
Logistic
Polynomial
Naive Bayes
D-tree
SVM
Error
Linear
Logistic
Polynomial
Naive Bayes
D-tree
SVM