Title - Servo Prediction using Linear Regression Model

A Servo Prediction model, also known as a servo control or servo system, is a control system that uses feedback to accurately position or control the motion of a mechanical device, such as a motor or an actuator. The goal of a servo system is to maintain a desired position or trajectory by continuously monitoring the actual position and making adjustments as needed.

In Python, you can develop a servo prediction model using various libraries and techniques.

Objective

Objective of Servo prediction model is to predict Class of a vehicle based on its Motor, Screw, Pgain & Vgain.

Data Source

The dataset was taken from Kaggle which provides various kinds of dataset for projects.

Attributes in the dataset are -

Motor

Screw

Pgain

Vgain

Class

Import the library

In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

Import CSV as DataFrame

In [3]:
```
df=pd.read_csv(r'C:\Users\lahar\Documents\YBI internship servo prediction.csv')
```

Get the First Five Rows of Dataframe

In [4]:
```
df.head()
```

Out[4]:

	Motor	Screw	Pgain	Vgain	Class
0	E	E	5	4	4
1	B	D	6	5	11
2	D	D	4	3	6
3	B	A	3	2	48
4	D	B	6	5	6

Get Information of Dataframe

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Motor   167 non-null    object
 1   Screw   167 non-null    object
 2   Pgain   167 non-null    int64 
 3   Vgain   167 non-null    int64 
 4   Class   167 non-null    int64 
dtypes: int64(3), object(2)
memory usage: 6.7+ KB
```

Get the Summary Statistics

In [6]:
```
df.describe()
```

Out[6]:

	Pgain	Vgain	Class
count	167.000000	167.000000	167.000000
mean	4.155689	2.538922	21.173653
std	1.017770	1.369850	13.908038
min	3.000000	1.000000	1.000000
25%	3.000000	1.000000	10.500000
50%	4.000000	2.000000	18.000000
75%	5.000000	4.000000	33.500000
max	6.000000	5.000000	51.000000

In [7]:
```
df.nunique()
```

Out[7]:
```
Motor     5
Screw     5
Pgain     4
Vgain     5
Class    51
dtype: int64
```

Get Column Names

In [8]:
```
df.columns
```
Out[8]:
```
Index(['Motor', 'Screw', 'Pgain', 'Vgain', 'Class'], dtype='object')
```

Get Shape of Dataframe

In [9]:
```
df.shape
```
Out[9]:
```
(167, 5)
```

Get Categories and Counts of Categorical Variables

In [10]:
```
df[['Motor']].value_counts()
```
Out[10]:
```
Motor
C        40
A        36
B        36
E        33
D        22
dtype: int64
```

In [11]:
```
df[['Screw']].value_counts()
```
Out[11]:
```
Screw
A        42
B        35
C        31
D        30
E        29
dtype: int64
```

Get Encoding of Categorical Features

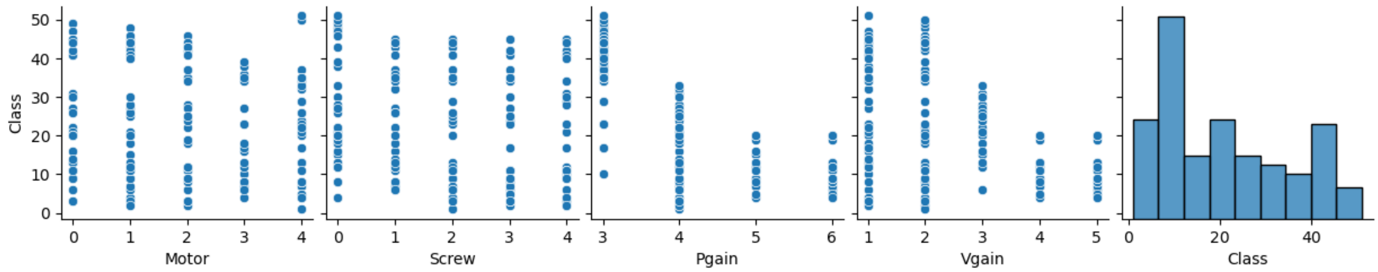
In [12]:
```
df.replace({'Motor':{'A':0,'B':1,'C':2,'D':3,'E':4}},inplace=True)
```

In [13]:
```
df.replace({'Screw':{'A':0,'B':1,'C':2,'D':3,'E':4}},inplace=True)
```

Data Visualization

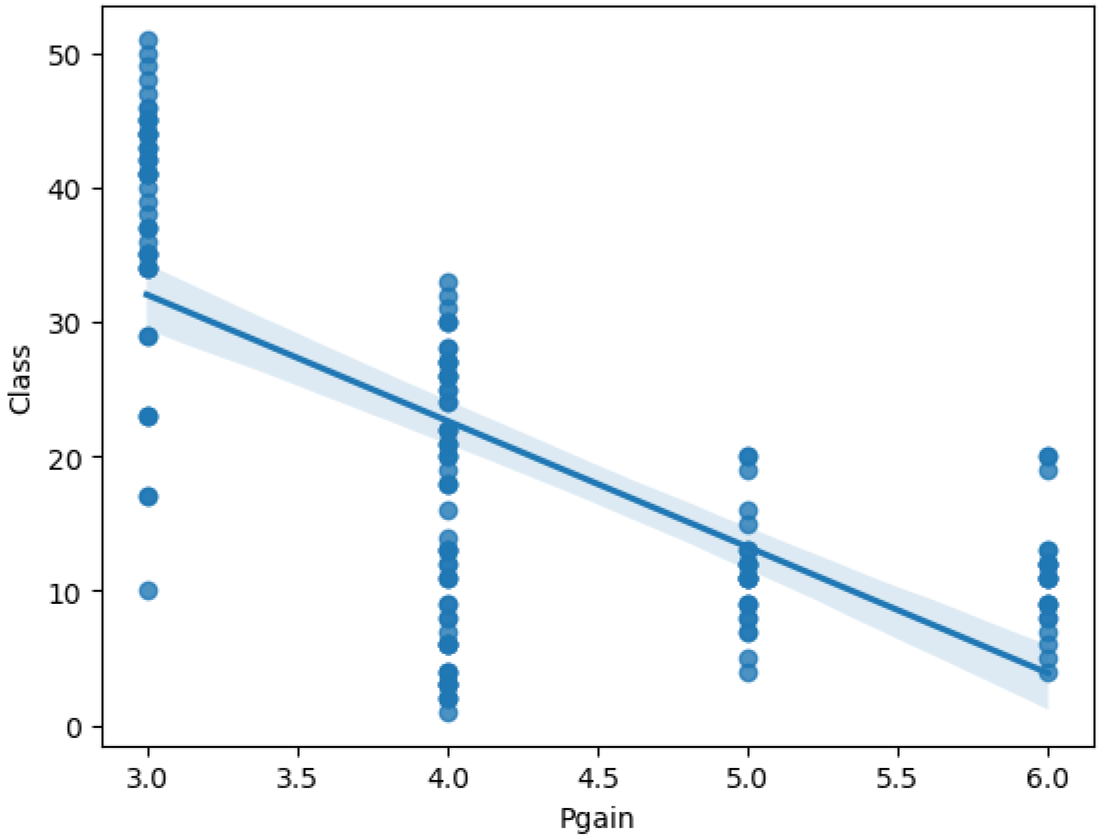
In [14]:
```
import seaborn as sns
```

In [15]:
```
sns.pairplot(df, x_vars=['Motor','Screw','Pgain','Vgain','Class'],y_vars=['Class'])
```
Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x1278bf1d550>
```



In [16]:
```python
sns.regplot(x='Pgain',y='Class',data=df)
```

Out[16]: <Axes: xlabel='Pgain', ylabel='Class'>



Data Preprocessing

In [17]:
```python
df.corr()
```

Out[17]:

	Motor	Screw	Pgain	Vgain	Class
Motor	1.000000	-0.052501	-0.037214	-0.003801	-0.112938
Screw	-0.052501	1.000000	-0.099503	0.011336	-0.162240
Pgain	-0.037214	-0.099503	1.000000	0.812268	-0.687098
Vgain	-0.003801	0.011336	0.812268	1.000000	-0.391963
Class	-0.112938	-0.162240	-0.687098	-0.391963	1.000000

Remove Missing Values

In [18]:
```python
df=df.dropna()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Motor   167 non-null    int64
 1   Screw   167 non-null    int64
 2   Pgain   167 non-null    int64
 3   Vgain   167 non-null    int64
 4   Class   167 non-null    int64
dtypes: int64(5)
memory usage: 6.7 KB
```

Define y(dependent or label or target variable) and X(independent or features or attribute variable)

In [20]:
```
y = df['Class']
```

In [21]:
```
y.shape
```

Out[21]:
```
(167,)
```

In [22]:
```
y
```

Out[22]:
```
0       4
1      11
2       6
3      48
4       6
       ..
162    44
163    40
164    25
165    44
166    20
Name: Class, Length: 167, dtype: int64
```

In [23]:
```
X = df[['Motor','Screw','Pgain','Vgain']]
```

In [24]:
```
X.shape
```

Out[24]:
```
(167, 4)
```

In [25]:
```
X
```

Out[25]:

	Motor	Screw	Pgain	Vgain
0	4	4	5	4
1	1	3	6	5
2	3	3	4	3
3	1	0	3	2
4	3	1	6	5
...	...	...	...	...
162	1	2	3	2
163	1	4	3	1
164	2	3	4	3
165	0	1	3	2
166	0	0	6	5

167 rows × 4 columns

Get Train Test Split

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
X_train, X_test, y_train , y_test = train_test_split(X,y, test_size=0.3, random_state=25
```

In [28]:
```
X_train.shape, X_test.shape, y_train.shape , y_test.shape
```

Out[28]:
```
((116, 4), (51, 4), (116,), (51,))
```

Get Model Train

In [29]:
```
from sklearn.linear_model import LinearRegression
```

In [30]:
```
lr=LinearRegression()
```

In [31]:
```
lr.fit(X_train,y_train)
```

Out[31]:
```
▼ LinearRegression
LinearRegression()
```

Get Model Prediction

In [32]:
```
y_pred = lr.predict(X_test)
```

In [33]:
```
y_pred.shape
```

Out[33]:
```
(51,)
```

In [34]:
```
y_pred
```

Out[34]:
```
array([24.55945258, 30.98765106, 18.54485477, 25.51524243, 38.56082023,
       23.52007775, 11.61947065, 20.03335614, 40.60404401, 41.7009556 ,
       13.66269443, 26.01242807, 16.50163099, 16.54663453, 21.92598051,
       22.52570646, -5.46449561, 30.68912392, 32.7323477 ,  1.41282941,
       33.97718702, 31.63543611, 33.52806048, 30.04133887, 19.38557109,
        6.49364826, 28.5528375 , 17.04382017, 25.06611589,  3.50411229,
       30.59606128, 23.67067716, 35.72188367, 32.08456265, 12.46018697,
        3.6547117 , 23.47201865, 33.03087484, 17.49294672, 37.61450804,
       27.54898855, 22.07657992, 11.51387478,  9.470651  , 30.53852451,
       28.64590014, 33.67865989,  4.60102388, 24.1198037 , 21.13026773,
       25.71390094])
```

Get Model Evaluation

In [35]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [36]:
```
mean_squared_error(y_test,y_pred)
```

Out[36]:
```
66.03589175595567
```

In [37]:
```
mean_absolute_error(y_test,y_pred)
```

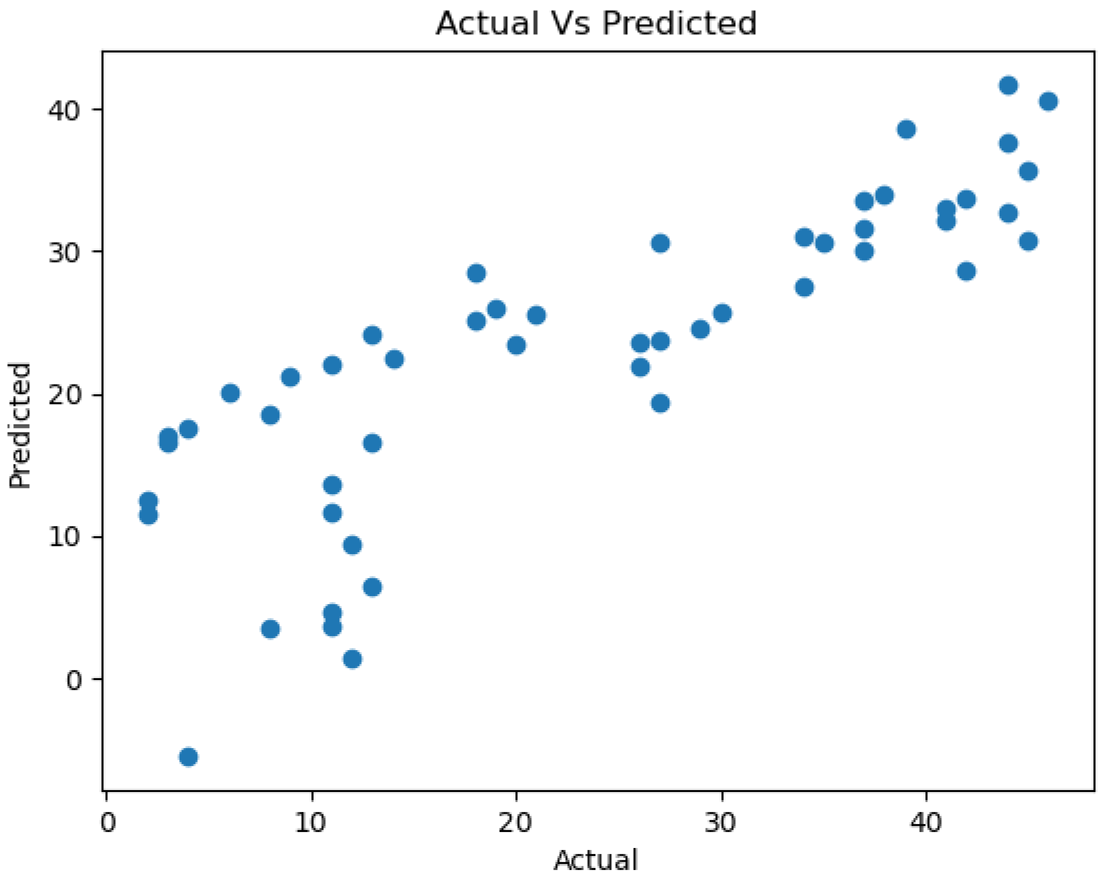
Out[37]:
```
7.190539677251239
```

In [38]:
```
r2_score(y_test,y_pred)
```

Out[38]:
```
0.6807245170563925
```

Get Visualisation of Actual vs Predicted Results

In [39]:
```
import matplotlib.pyplot as plt
plt.scatter(y_test,y_pred)
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual Vs Predicted")
plt.show()
```



Get Future Predictions

In [40]:
```
X_new = df.sample(1)
```

In [41]:
```
X_new
```

Out[41]:

	Motor	Screw	Pgain	Vgain	Class
130	4	3	5	4	5

In [42]:
```
X_new.shape
```

Out[42]:
```
(1, 5)
```

In [43]:
```
X_new = X_new.drop('Class',axis=1)
```

In [44]:
```
X_new
```

Out[44]:

	Motor	Screw	Pgain	Vgain
130	4	3	5	4

In [45]:
```
X_new.shape
```

Out[45]:
```
(1, 4)
```

In [46]:
```
y_pred_new = lr.predict(X_new)
```

In [47]:
```
y_pred_new
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Out[47]:
```
array([7.53302309])
```

Explaination

The Servo Prediction model is a control system that accurately positions or controls the motion of a mechanical device, such as a motor or actuator. It uses feedback to maintain a desired position or trajectory.

Python offers several libraries for building servo prediction models, including TensorFlow, Keras, PyTorch, and scikit-learn. These libraries provide the necessary tools and functions for data preprocessing, model training, and evaluation.

Building an accurate servo prediction model may involve an iterative process of collecting data, training the model, and evaluating its performance to refine and improve it.

In [ ]: