Steps for Regression Model with SKlearn

Import Library

```
import pandas as pd
```

```
import numpy as np
```

Import CSV as DataFrame

```
df = pd.read_csv('https://github.com/YBI-Foundation/Dataset/raw/main/Fish.csv')
```

Get first 5 rows

```
df.head()
```

	Category	Species	Weight	Height	Width	Length1	Length2	Length3
0	1	Bream	242.0	11.5200	4.0200	23.2	25.4	30.0
1	1	Bream	290.0	12.4800	4.3056	24.0	26.3	31.2
2	1	Bream	340.0	12.3778	4.6961	23.9	26.5	31.1
3	1	Bream	363.0	12.7300	4.4555	26.3	29.0	33.5
4	1	Bream	430.0	12.4440	5.1340	26.5	29.0	34.0

Get information of the dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Category  159 non-null    int64  
 1   Species   159 non-null    object 
 2   Weight    159 non-null    float64
 3   Height    159 non-null    float64
 4   Width     159 non-null    float64
 5   Length1   159 non-null    float64
 6   Length2   159 non-null    float64
 7   Length3   159 non-null    float64
dtypes: float64(6), int64(1), object(1)
memory usage: 10.1+ KB
```

Get the summary statistics

```
df.describe()
```

	Category	Weight	Height	Width	Length1	Length2	Length3
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	3.264151	398.326415	8.970994	4.417486	26.247170	28.415723	31.227044
std	1.704249	357.978317	4.286208	1.685804	9.996441	10.716328	11.610246
min	1.000000	0.000000	1.728400	1.047600	7.500000	8.400000	8.800000
25%	2.000000	120.000000	5.944800	3.385650	19.050000	21.000000	23.150000
50%	3.000000	273.000000	7.786000	4.248500	25.200000	27.300000	29.400000
75%	4.500000	650.000000	12.365900	5.584500	32.700000	35.500000	39.650000
max	7.000000	1650.000000	18.957000	8.142000	59.000000	63.400000	68.000000

Get shape of Dataframe

```
df.shape
```

```
(159, 8)
```

Get column names

```
df.columns
```

```
Index(['Category', 'Species', 'Weight', 'Height', 'Width', 'Length1',
       'Length2', 'Length3'],
      dtype='object')
```

```
y = df['Weight']
```

```
y.shape
```

```
(159,)
```

```
y
```

```
0      242.0
1      290.0
2      340.0
3      363.0
4      430.0
       ...
```

```
154     12.2
155     13.4
156     12.2
157     19.7
158     19.9
Name: Weight, Length: 159, dtype: float64
```

```
X=df[['Height', 'Width', 'Length1',
       'Length2', 'Length3']]
```

```
X = df.drop(['Category', 'Species', 'Weight'], axis = 1)
```

```
X.shape
```

```
(159, 5)
```

```
X
```

	Height	Width	Length1	Length2	Length3
0	11.5200	4.0200	23.2	25.4	30.0
1	12.4800	4.3056	24.0	26.3	31.2
2	12.3778	4.6961	23.9	26.5	31.1
3	12.7300	4.4555	26.3	29.0	33.5
4	12.4440	5.1340	26.5	29.0	34.0
...	...	...	...	...	...
154	2.0904	1.3936	11.5	12.2	13.4
155	2.4300	1.2690	11.7	12.4	13.5
156	2.2770	1.2558	12.1	13.0	13.8
157	2.8728	2.0672	13.2	14.3	15.2
158	2.9322	1.8792	13.8	15.0	16.2

159 rows × 5 columns

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 2529)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((111, 5), (48, 5), (111,), (48,))
```

Get Model Train

You can choose machine learning models as per requirement

from sklearn.linear_model import LinearRegression

from sklearn.linear_model import Ridge from sklearn.linear_model import Lasso from sklearn.linear_model import ElasticNet from sklearn.linear_model import PoissonRegressor from sklearn.linear_model import TweedieRegressor from sklearn.linear_model import GammaRegressor from sklearn.neighbors import BayesianRegressor from sklearn.svm import KNeighborsRegressor from sklearn.tree import SVR from sklearn.ensemble import DecisionTreeRegressor from sklearn.ensemble import RandomForestRegressor from sklearn.ensemble import StackingRegressor from sklearn.ensemble import BaggingRegressor from sklearn.ensemble import VotingRegressor from sklearn.ensemble import HistGradientBoostingRegressor from sklearn.ensemble import AdaBoostRegressor from sklearn.ensemble import GradientBoostingRegressor

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
     LinearRegression()
```

Get Model Prediction

```
y_pred = model.predict(X_test)
```

```
y_pred.shape
```

```
     (48,)
```

```
y_pred
```

```
     array([ 485.76826299,  502.24720857,   94.72381964,  876.5711712 ,
             184.0789176 ,  219.30130488,  322.32532246,  376.22325991,
             372.35730485, -182.67537078, -160.60486837,  454.33586185,
             159.59755829,  843.48525226,  587.21680573,  299.53521445,
             597.72950823,  197.14605397,  639.89046741,   91.20067876,
             150.95424753, -103.08320574,  627.19712753,  795.69176861,
             814.68732975, -204.1496511 ,  329.98746856,  715.89288013,
             359.75634357,  792.3243925 ,  532.7036706 ,  552.00832342,
             433.48472727,  687.61750267, -204.76362537,  932.53668294,
             810.74234216,  -80.06217174,  284.36287887,  907.08036021,
             642.5828335 ,  959.33848223,  675.28792291,  718.86305458,
             623.89849226,  376.48346981,  530.83828119,  -86.2357066 ])
```

Get Model Evaluation

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error,
```

```
mean_squared_error(y_test, y_pred)
```

```
16397.344524411383
```

```
mean_absolute_error(y_test, y_pred)
```

```
103.02952922678541
```

```
mean_absolute_percentage_error(y_test, y_pred)
```

```
2.5082853471600237
```

```
r2_score(y_test, y_pred)
```

```
0.8349141424416877
```

Get Future Predictions

steps to follow

1. Extract a random row using sample function
2. Separate X and y
3. Predict

```
df_new = df.sample(1)
```

```
df_new
```

	Category	Species	Weight	Height	Width	Length1	Length2	Length3
134	4	Pike	456.0	7.28	4.3225	40.0	42.5	45.5

```
X_new = df_new[['Height', 'Width', 'Length1','Length2', 'Length3']]
```

```
X_new.shape
```

```
(1, 5)
```

```
y_pred_new = model.predict(X_new)
```

```
y_pred_new
```

```
array([759.18048356])
```