Ishika kesarwani roll no. 92 SAP ID 500076373

Pattern and Anomaly Detection Lab 1

Installing Anaconda and setup up environment

Steps in building regression model

1. Collect/extract data
2. Pre-process it.
3. Creating train and test datasets
4. Visualization and descriptive analytics of patterns present in the data
5. Model building (simple linear regression)
6. Validation and evaluation of model.

Population Y intercept
Population Slope Coefficient
Independent Variable
Random Error term
Dependent Variable

$$Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x};$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2},$$

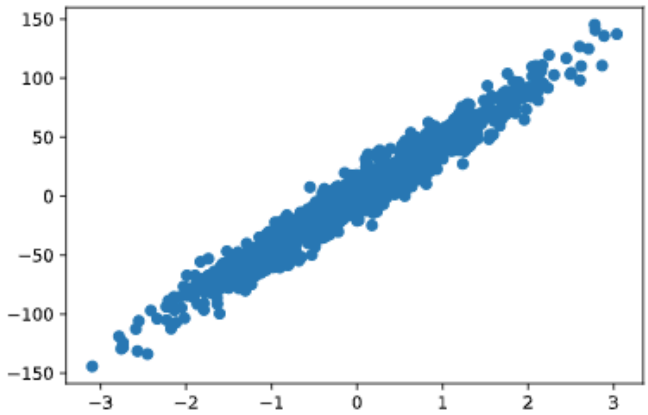
In [69]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [70]:
```
#create a dataset using sklearn x scalar,y scalar,N = 1000
from sklearn.datasets import make_regression
x,y = make_regression(n_samples=1000,n_features=1,noise=10,random_state=101)
```

In [71]:
```
import numpy as np
#print mean, standar deviation, and variance of x
print(np.mean(x))
print(np.std(x))
print(np.var(x))
```

```
0.026432601209742754
1.0529048585460918
1.1086086411499658
```

In [72]:
```
#plot the data
plt.scatter(x,y)
plt.show()
```



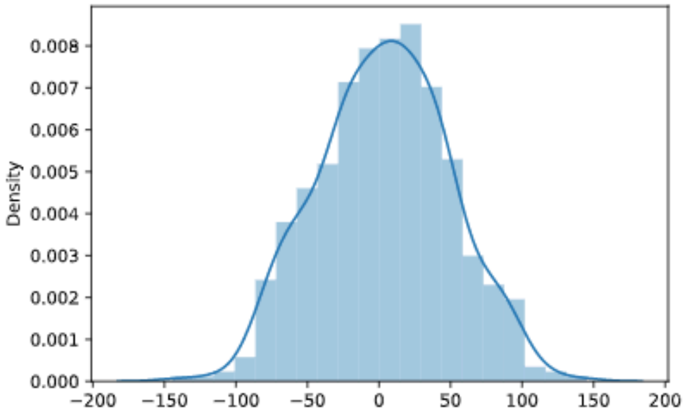
In [73]:
```
#preprocess data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x)
x = scaler.transform(x)
```

In [74]:
```
#print mean, standar deviation, and variance of x
print(np.mean(x))
print(np.std(x))
print(np.var(x))
```

```
1.5987211554602253e-17
1.0000000000000002
1.0000000000000004
```

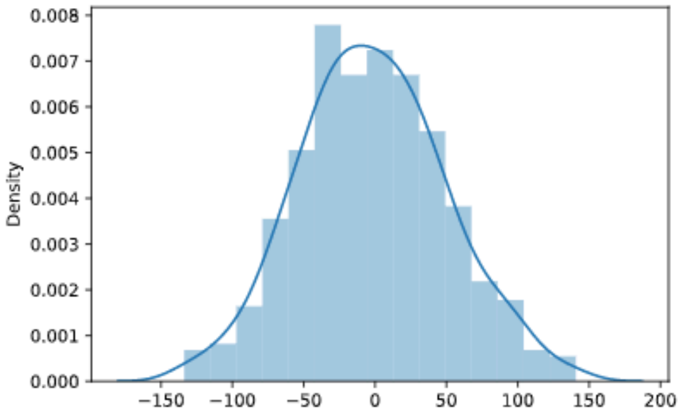
In [56]:
```
#create traing and test sets
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.4,random_state=101)
```

In [57]:
```
#visualization and descriptive analytics of patterns present in the data
sns.distplot(y_train)
plt.show()
```



In [58]:
```
#visualization and descriptive analytics of patterns present in the data
sns.distplot(y_test)
plt.show()
```

```
/Users/ishikakesarwani/opt/anaconda3/lib/python3.8/site-packages/seaborn/distributions.py:2557: FutureWarning: `dist
plot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [59]:
```
#Model building (simple linear regression)
from sklearn.linear_model import LinearRegression
#instantiate an instance of linear regression model
model = LinearRegression()
#inplace of passing X and y, we pass X_train & y_train
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
```

In [77]:
```
print(model.coef_)
```

```
[46.72443049]
```

In [78]:
```
print(model.intercept_)
```

```
1.4670708447023015
```

There are 3 common evaluation metrices for regression problem 1 Mean Absolute Error

2 Mean squared Error

3 Root Mean squared Error

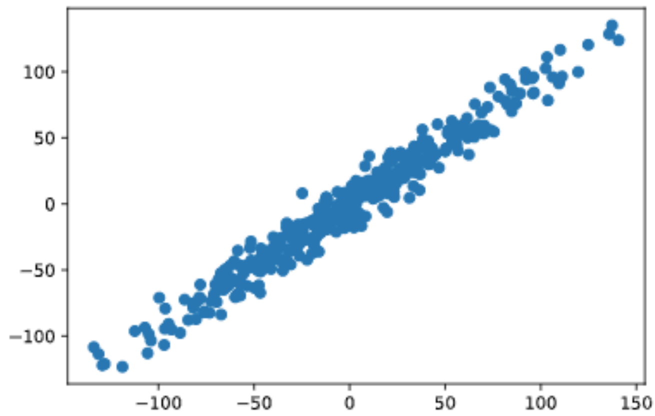
All of these are **Loss function** , but we want to minimize them

In [60]:
```
#validation and evaluation of model
from sklearn import metrics
print(f"Mean Absolute Error is => {metrics.mean_absolute_error(y_test,y_pred)}")
print(f"Mean Squared Error is => {metrics.mean_squared_error(y_test,y_pred)}")
print(f"Root Mean Squared Error is => {np.sqrt(metrics.mean_squared_error(y_test,y_pred))}")
```

```
Mean Absolute Error is => 7.908835058320017
Mean Squared Error is => 98.05110113937094
Root Mean Squared Error is => 9.902075597538676
```

In [61]:
```
#finding how far predictions is diff from y_test
plt.scatter(y_test,y_pred)
```

Out[61]: <matplotlib.collections.PathCollection at 0x7f9fb96e17c0>



In [62]:
```
#residual=diff in between actual values(y_test) & predicted values
sns.histplot((y_test-y_pred))
```

Out[62]: <AxesSubplot:ylabel='Count'>