CODE:

In [1]:

```
import numpy as np
class PerceptronRegressor:
    def __init__(self, learning_rate=0.01, n_iterations=100):
        self.learning_rate = learning_rate
        self.n_iterations = n_iterations
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.n_iterations):
            for i in range(n_samples):
                y_predicted = np.dot(X[i], self.weights) + self.bias
                error = y[i] - y_predicted
                self.weights += self.learning_rate * error * X[i]
                self.bias += self.learning_rate * error

    def predict(self, X):
        return np.dot(X, self.weights) + self.bias
```

In [2]:

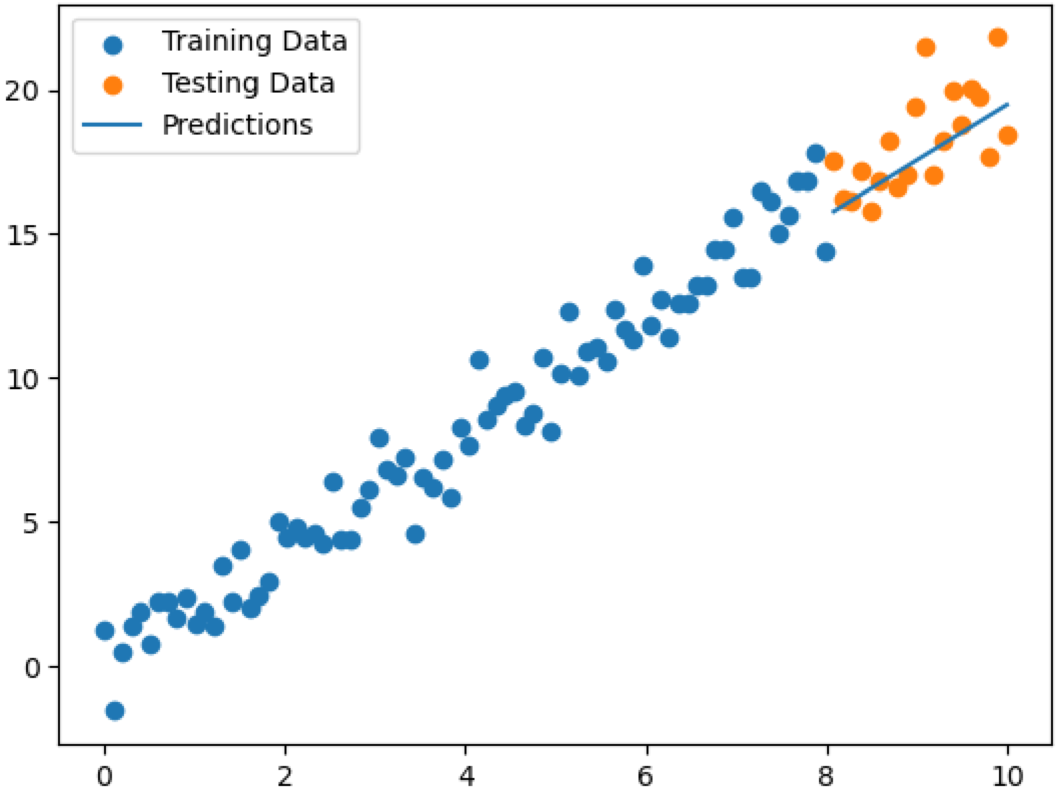
```
import matplotlib.pyplot as plt
X = np.linspace(0, 10, 100)
y = X * 2 + np.random.normal(size=100)
X_train, X_test = X[:80], X[80:]
y_train, y_test = y[:80], y[80:]
```

In [3]:

```
perceptron = PerceptronRegressor(learning_rate=0.01, n_iterations=100)
perceptron.fit(X_train.reshape(-1, 1), y_train)
y_pred = perceptron.predict(X_test.reshape(-1, 1))
```

In [4]:

```
plt.scatter(X_train, y_train, label='Training Data')
plt.scatter(X_test, y_test, label='Testing Data')
plt.plot(X_test, y_pred, label='Predictions')
plt.legend()
plt.show()
```



```
In [ ]:
```

RESULT:

Hence, we successfully implemented Single Layer Perceptron for Regression Problem.