In [57]:
```
import numpy as np
import matplotlib.pyplot as plt
import csv
import pandas as pd
np.random.seed(2023)

# Load data
df = pd.read_csv('ford-prices.csv')
df = df.replace({',': ''}, regex=True)
df = df.astype(float)
print(df.head())
```

```
     Year  minPrice  maxPrice
0  1992.0    8730.0   14840.0
1  1993.0    8781.0   16535.0
2  1994.0    9449.0   18328.0
3  1995.0   10224.0   19571.0
4  1996.0   10575.0   20295.0
```

In [58]:
```
# Split data into features (years) and labels (minimum prices and maximum pr
X = df.iloc[:,0].values.reshape(-1,1)
y_min = df.iloc[:,1].values.reshape(-1,1)
y_max = df.iloc[:,2].values.reshape(-1,1)
```

In [59]:
```
# Initialize theta
theta_min = np.random.rand(2,1)
theta_max = np.random.rand(2,1)
```

In [60]:
```
# Define the hypothesis function
def h(X, theta):
    return np.matmul(X, theta)
```

In [61]:
```
# Define the mean squared error loss function
def MSE(y, y_hat):
    return np.mean((y-y_hat)**2)
```

In [62]:
```
def gradient_descent(X, y, theta, alpha, num_iters):
    m = X.shape[0]
    loss = np.zeros((num_iters, 1))
    for i in range(num_iters):
        y_hat = h(X, theta)
#         print(y_hat.shape)
        loss[i] = MSE(y, y_hat)
        #print(loss[i])
#         theta = theta + (alpha)/m * np.matmul(X.T, y- y_hat)
        error = y_hat - y
        theta[0] -= alpha * error.mean()
        theta[1] -= alpha * (error * X[:,1:]).mean()

    return theta, loss
```

In [63]:
```
# Set hyperparameters
alpha = 1e-7
```

```
num_iters = 100
```

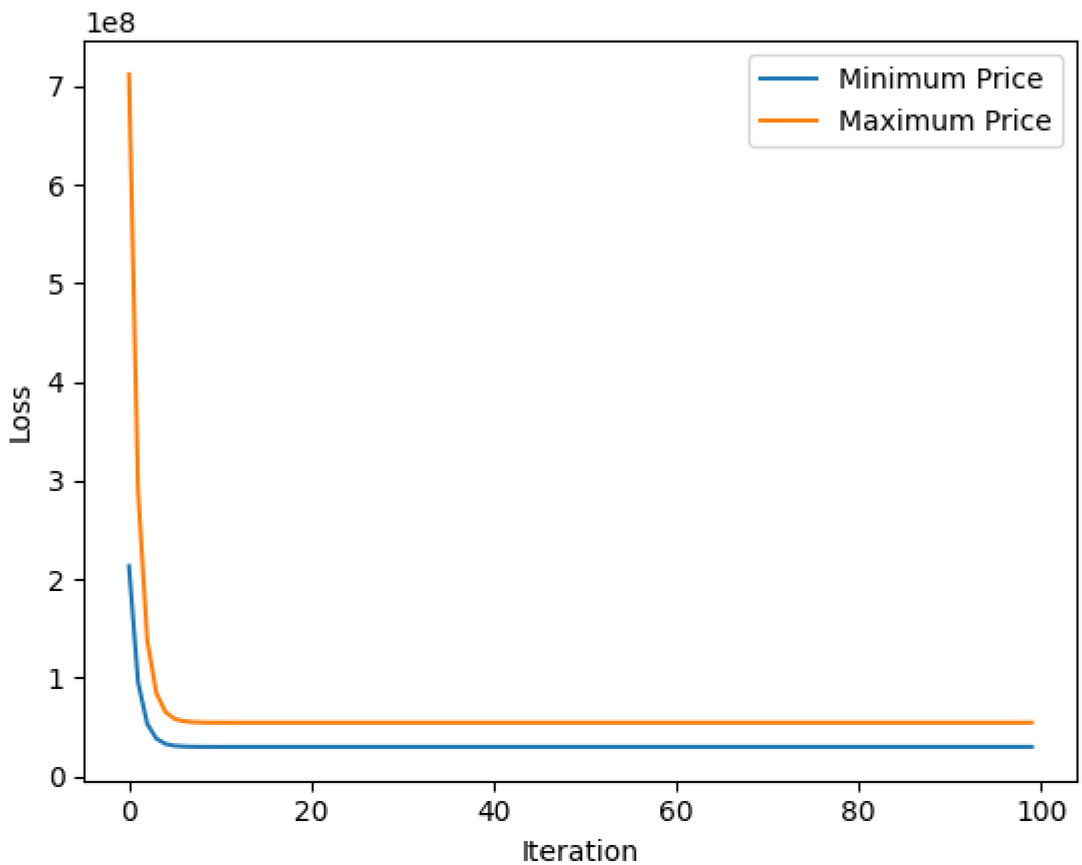
In [64]:
```
# Run gradient descent for minimum prices
theta_min, loss_min = gradient_descent(np.hstack((np.ones((X.shape[0], 1)),

# Run gradient descent for maximum prices
theta_max, loss_max = gradient_descent(np.hstack((np.ones((X.shape[0], 1)),
```

In [65]:
```
print(f'The values of theta for the minimum prices are : theta0 {theta_min[0
print(f'The values of theta for the maximum prices are : theta0 {theta_max[0
```

```
The values of theta for the minimum prices are : theta0 [0.32510465],theta1
[7.64557759]
The values of theta for the maximum prices are : theta0 [0.59409396],theta1
[12.91118457]
```

In [66]:
```
# Plot the loss curve
plt.plot(range(num_iters), loss_min, label='Minimum Price')
plt.plot(range(num_iters), loss_max, label='Maximum Price')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

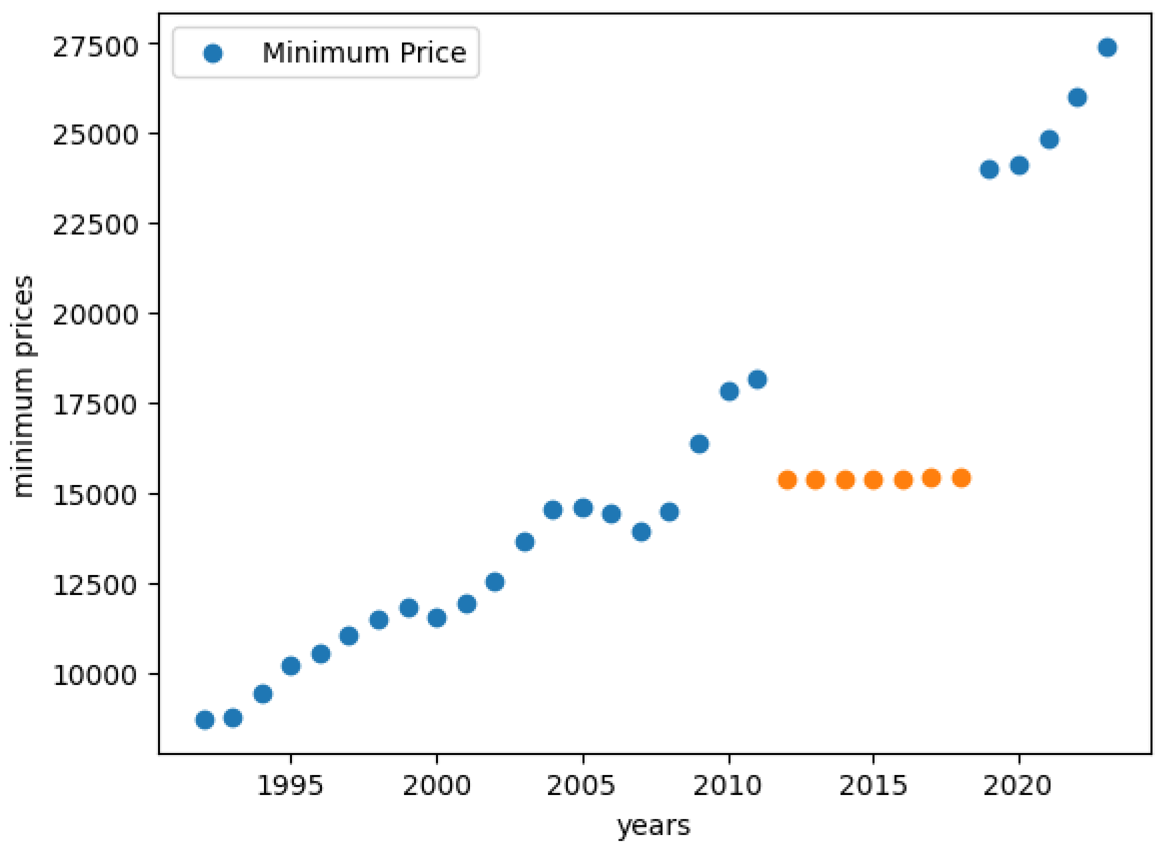


predicting the minimum and maximum price of year between 2012 to 2018

In [67]:
```
prices_min = []
prices_max = []
years = []
for i in range(2012,2019):
    years.append(i)
    prices_min.append(h(np.array([1,i]), theta_min))
    prices_max.append(h(np.array([1,i]), theta_max))
```

plotting the Minimum prices of every years

In [68]:
```
plt.scatter(X, y_min, label='Minimum Price')
plt.scatter(np.array(years),np.array(prices_min))
plt.xlabel('years')
plt.ylabel('minimum prices')
plt.legend()
plt.show()
```



plotting the Maximum prices of every years

In [69]:
```
plt.scatter(X, y_max, label='Maximum Price')
plt.scatter(np.array(years),np.array(prices_max))
plt.xlabel('years')
plt.ylabel('maximum prices')
plt.legend()
plt.show()
```

dynamic learning rate with feature scaling

In [70]:
```
X_scaled = (X - X.mean())/(X.max()-X.min())
y_min_scaled = (y_min - y_min.mean())/(y_min.max()-y_min.min())
y_max_scaled = (y_max - y_max.mean())/(y_max.max()-y_max.min())
```

In [71]:
```
# Initialize theta
theta_min = np.random.rand(2,1)
theta_max = np.random.rand(2,1)
alpha = 1e-3
```

In [72]:
```
def gradient_descent_dynamic(X, y, theta, alpha, num_iters):
    m = X.shape[0]
    loss = np.zeros((num_iters, 1))
    for i in range(num_iters):
        y_hat = h(X, theta)
        loss[i] = MSE(y, y_hat)
        error = y_hat - y
        theta[0] -= alpha * error.mean()
        theta[1] -= (1/(1+i))*alpha * (error * X[:,1:]).mean()

    return theta, loss
```

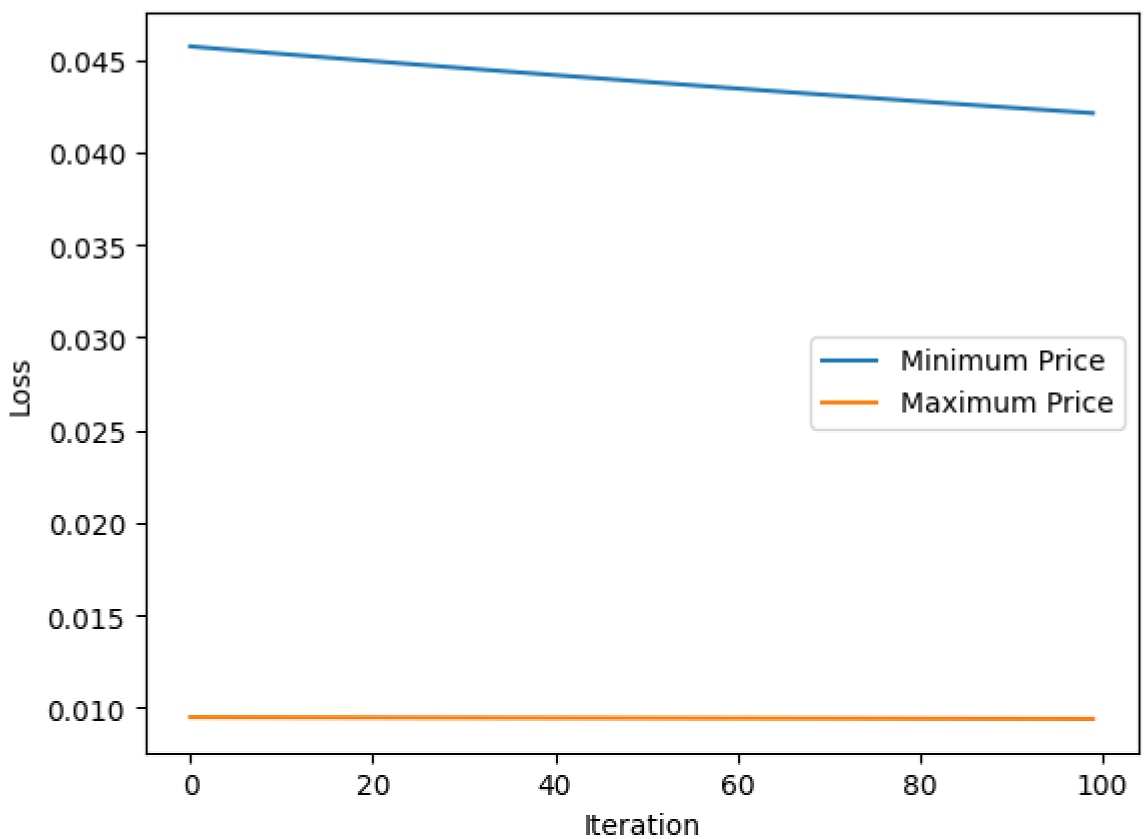
In [73]:
```
# Run gradient descent for minimum prices
theta_min, loss_min = gradient_descent_dynamic(np.hstack((np.ones((X_scaled.

# Run gradient descent for maximum prices
theta_max, loss_max = gradient_descent_dynamic(np.hstack((np.ones((X_scaled.
```

In [74]:
```python
print(f'The values of theta for the minimum prices are : theta0 {theta_min[0
print(f'The values of theta for the maximum prices are : theta0 {theta_max[0
```

```
The values of theta for the minimum prices are : theta0 [0.12788443],theta1
[0.46813276]
The values of theta for the maximum prices are : theta0 [0.01998655],theta1
[0.72736549]
```

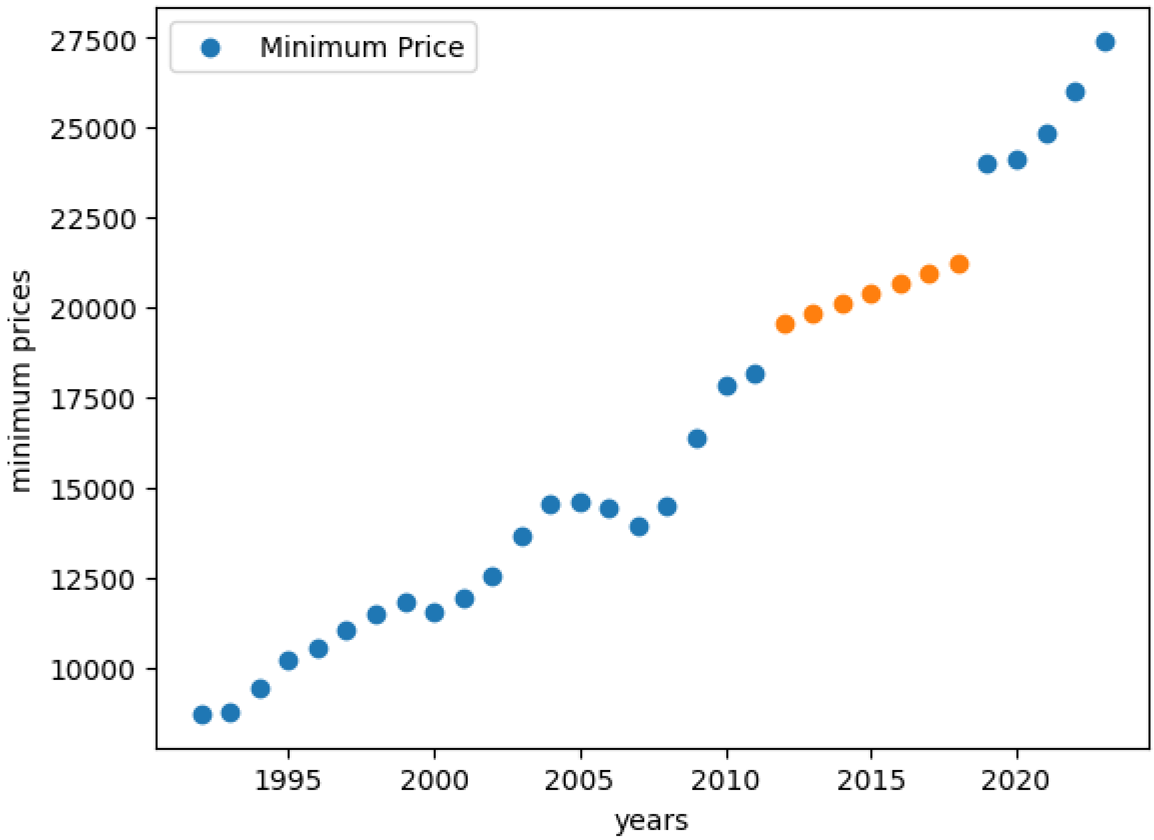
In [75]:
```python
# Plot the loss curve
plt.plot(range(num_iters), loss_min, label='Minimum Price')
plt.plot(range(num_iters), loss_max, label='Maximum Price')
plt.xlabel('Iteration')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [76]:
```python
prices_min = []
prices_max = []
years = []
for i in range(2012,2019):
    years.append(i)
    prices_min.append(h(np.array([1,(i-X.mean())/(X.max()-X.min())]), theta_
    prices_max.append(h(np.array([1,(i-X.mean())/(X.max()-X.min())]), theta_
```

In [77]:
```python
plt.scatter(X, y_min, label='Minimum Price')
plt.scatter(np.array(years),np.array(prices_min))
plt.xlabel('years')
plt.ylabel('minimum prices')
```

```
plt.legend()
plt.show()
```



In [78]:
```
plt.scatter(X, y_max, label='Maximum Price')
plt.scatter(np.array(years),np.array(prices_max))
plt.xlabel('years')
plt.ylabel('maximum prices')
plt.legend()
plt.show()
```

Summary

In [79]:
```
prices_min = h(np.array([1,(2024-X.mean())/(X.max()-X.min())]), theta_min)*(
prices_max = h(np.array([1,(2024-X.mean())/(X.max()-X.min())]), theta_max)*(
print(prices_min , prices_max)
```

```
[22939.18549421] [37773.83463379]
```

The probable maximum price for ford in 2024 is 37773.83463379 and minimum price would be 22939.18549421 which seems lower than 2023. So buying new card should be benificial.

The new values of theta for the minimum prices are : theta0 0.12788443,theta1 0.46813276 The values of theta for the maximum prices are : theta0 0.01998655,theta1 0.72736549 And form the price prediction plot it's pretty evident that our model is good for the prediction. So we can trust the new values of theta.

There are some hyper parameters like learning rate and number of iteration. We need to tune them to get better result. As we can tell from the loss curve , if we would have run the model for several epochs more, the model would have converged better.

We can also see the difference between the performance of non scaled feature models and scaled feature models. In the first two models where we didn't do feature scaling , they didn't converge properly. We can see that from the prediction results as well. But when we did the feature scaling the models converged really well. So we can confidently say feature scaling helped in this case.

I will definitely buy the next generation model for sure as they are more cost effiecient and the new model should come with new features as well.