Name: Srinivasan JP

Reg No: 21MIS1044

Linear Regression using Gradient descent method

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

Read the data set

In [ ]:
```
data = pd.read_csv('./tvmarketing.csv')
print(data.head())
```

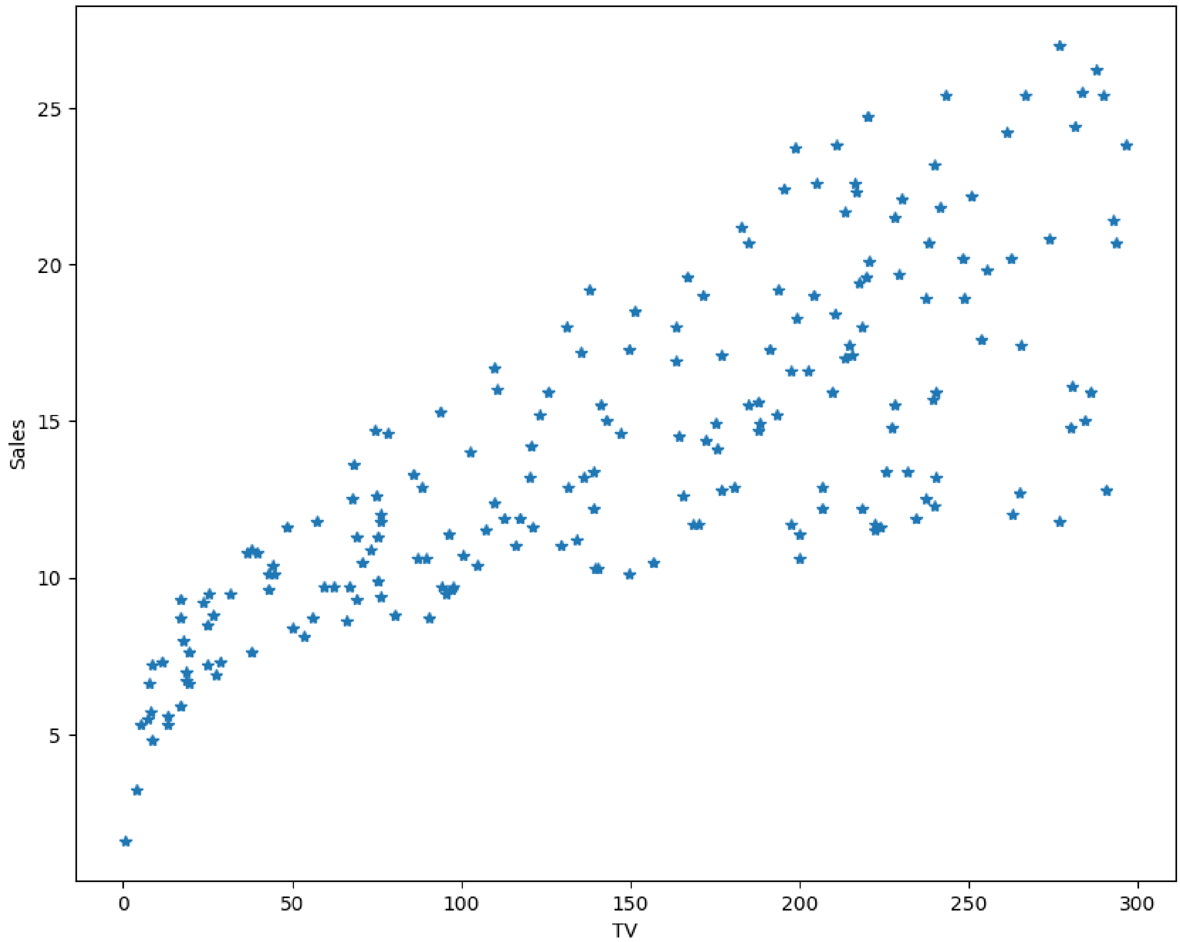
```
      TV  Sales
0  230.1   22.1
1   44.5   10.4
2   17.2    9.3
3  151.5   18.5
4  180.8   12.9
```

In [ ]:
```
X_df = pd.DataFrame(data.TV)
y_df = pd.DataFrame(data.Sales)
m = len(y_df)
```

Ploting the inital scatter plot to view the data points

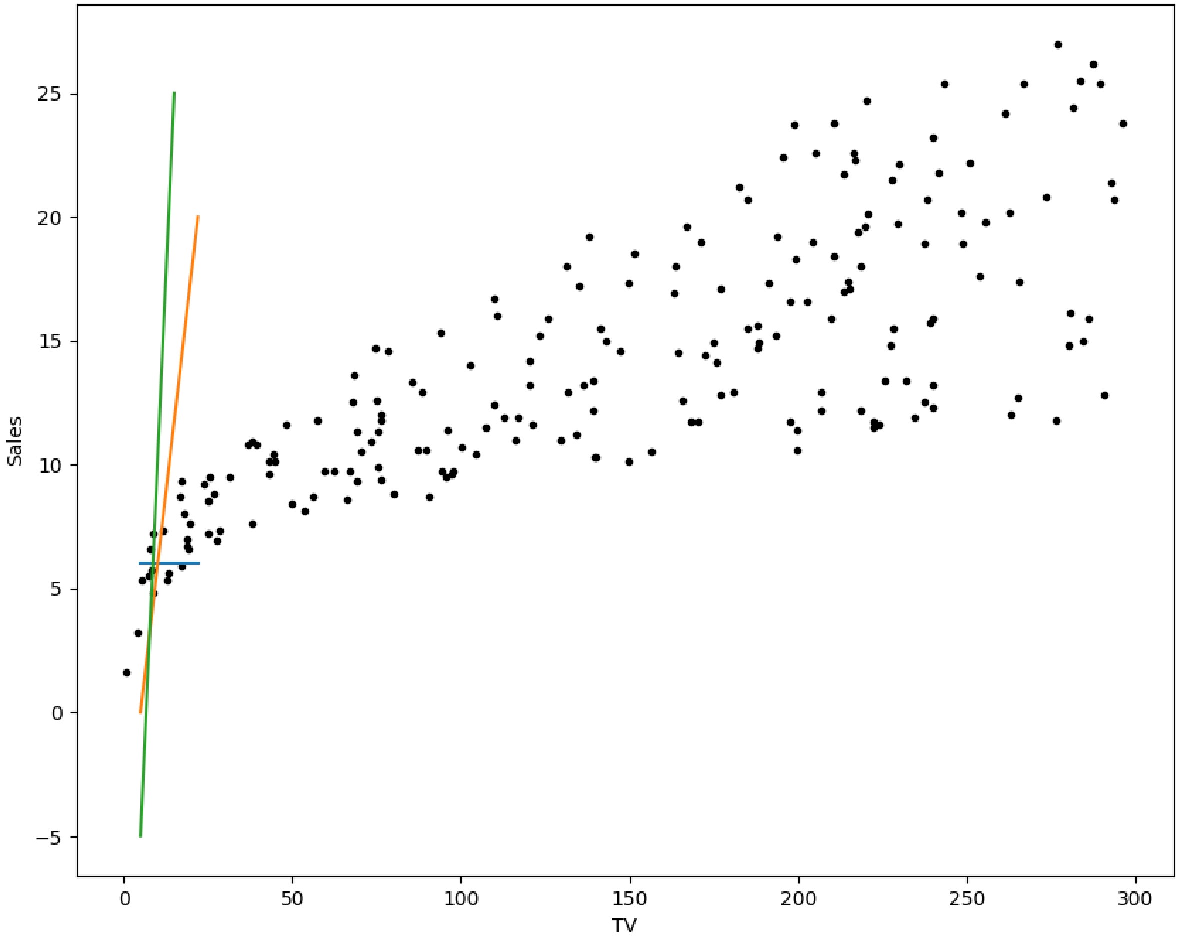
In [ ]:
```
plt.figure(figsize=(10,8))
plt.plot(X_df, y_df, '*')
plt.xlabel('TV')
plt.ylabel('Sales')
```

Out[ ]:
```
Text(0, 0.5, 'Sales')
```



In [ ]:
```
plt.figure(figsize=(10,8))
plt.plot(X_df, y_df, 'k.')
plt.plot([5, 22], [6,6], '-')
plt.plot([5, 22], [0,20], '-')
plt.plot([5, 15], [-5,25], '-')
plt.xlabel('TV')
plt.ylabel('Sales')
```

Out[ ]:
```
Text(0, 0.5, 'Sales')
```

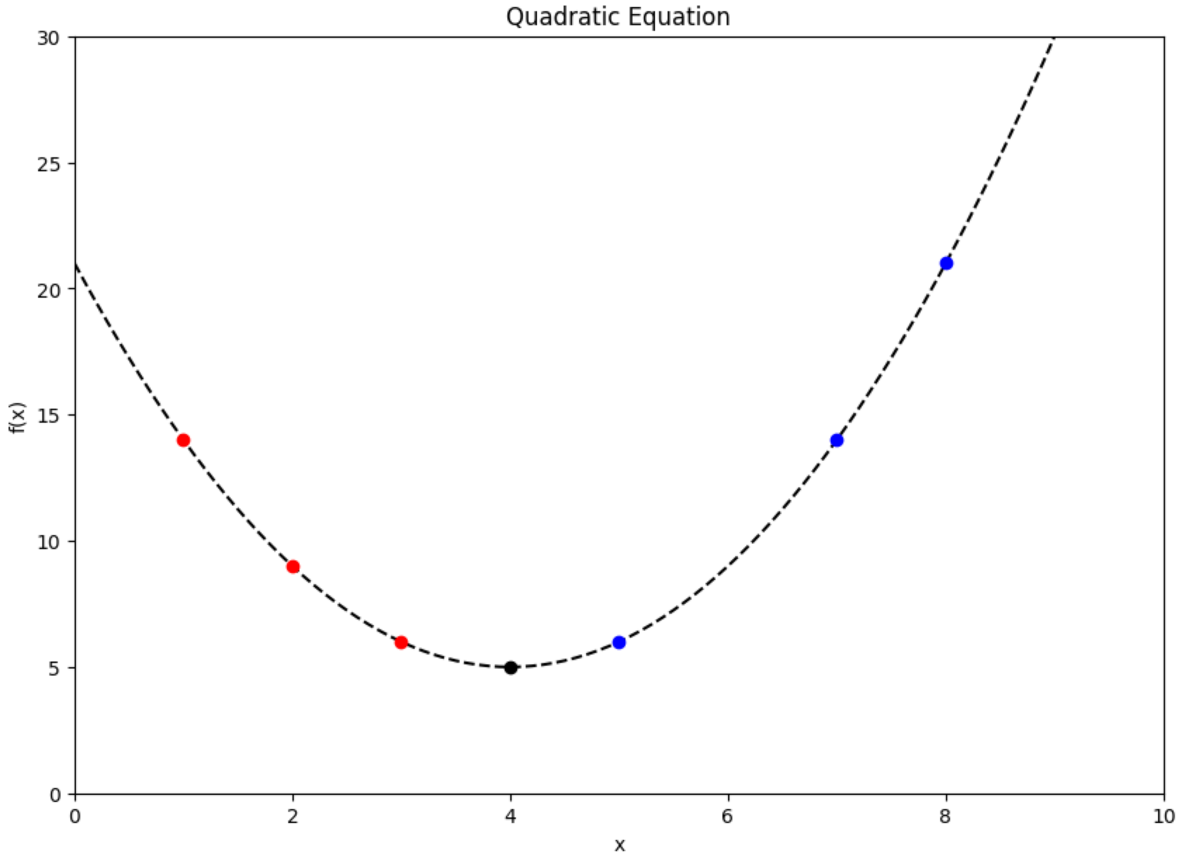


In [ ]:
```
x_quad = [n/10 for n in range(0, 100)]
y_quad = [(n-4)**2+5 for n in x_quad]
```

Ploting the quadratic equation where the Gradient point will move to find the best fit

In [ ]:
```
plt.figure(figsize = (10,7))
plt.plot(x_quad, y_quad, 'k--')
plt.axis([0,10,0,30])
plt.plot([1, 2, 3], [14, 9, 6], 'ro')
plt.plot([5, 7, 8],[6, 14, 21], 'bo')
plt.plot(4, 5, 'ko')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Quadratic Equation')
```

Out[ ]:
```
Text(0.5, 1.0, 'Quadratic Equation')
```



In [ ]:
```
iterations = 100
alpha = 0.01
```

In [ ]:
```
## Add a columns of 1s as intercept to X
X_df['intercept'] = 1

## Transform to Numpy arrays for easier matrix math and start theta at 0
X = np.array(X_df)
y = np.array(y_df).flatten()
theta = np.array([0, 0])
```

Function to calulate the cost function of the respective X,Y and theta

In [ ]:
```
def cost_function(X, y, theta):
    ## number of training examples
    m = len(y)

    ## Calculate the cost with the given parameters
    J = np.sum((X.dot(theta)-y)**2)/2/m

    return J
```

In [ ]:
```
cost_function(X, y, theta)
```

Out[ ]:
```
111.858125
```

Function to run the gradient iteration

In [ ]:
```
def gradient_descent(X, y, theta, alpha, iterations):

    cost_history = [0] * iterations

    for iteration in range(iterations):
```

```
        hypothesis = X.dot(theta)
        loss = hypothesis-y
        gradient = X.T.dot(loss)/m
        theta = theta - alpha*gradient
        cost = cost_function(X, y, theta)
        cost_history[iteration] = cost

    return theta, cost_history
```

In [ ]:

```
(t, c) = gradient_descent(X,y,theta,alpha, iterations)


## Print theta parameters
print (t)
print (np.array([3.5, 1]).dot(t))
print (np.array([7, 1]).dot(t))
```

```
[-8.80025526e+244 -4.46899119e+242]
-3.084558330655545e+245
-6.164647670123577e+245
```

finding the best fit among the theta

In [ ]:

```
## Plotting the best fit line
best_fit_x = np.linspace(0, 25, 20)
best_fit_y = [t[1] + t[0]*xx for xx in best_fit_x]
```

Ploting the best fit of the predicted item

In [ ]:

```
plt.figure(figsize=(10,6))
plt.plot(X_df.TV, y_df, '.')
plt.plot(best_fit_x, best_fit_y, '-')
plt.axis([0,25,-5,25])
plt.title('Sales vs. TV')
```

Out[ ]:

```
Text(0.5, 1.0, 'Sales vs. TV')
```