Introduction to Machine Learning - Linear Regression

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
sal = pd.read_csv('SalaryData.csv')
```

In [3]:
```
sal
```

Out[3]:

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891
5	2.9	56642
6	3.0	60150
7	3.2	54445
8	3.2	64445
9	3.7	57189
10	3.9	63218
11	4.0	55794
12	4.0	56957
13	4.1	57081
14	4.5	61111
15	4.9	67938
16	5.1	66029
17	5.3	83088
18	5.9	81363
19	6.0	93940
20	6.8	91738
21	7.1	98273
22	7.9	101302
23	8.2	113812
24	8.7	109431
25	9.0	105582
26	9.5	116969
27	9.6	112635
28	10.3	122391
29	10.5	121872

Data Exploration

In [10]:
```
X = sal['YearsExperience'] #feature
y = sal['Salary'] #target
```

In [5]:
```
X
```

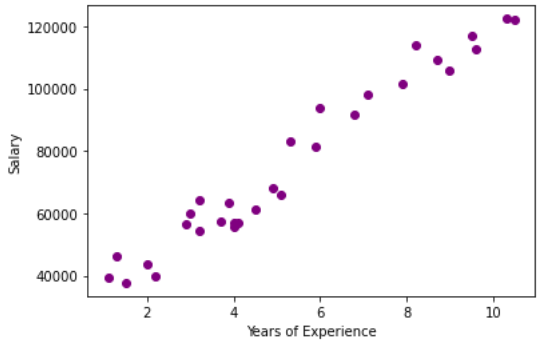
Out[5]:
```
0      1.1
1      1.3
2      1.5
3      2.0
4      2.2
5      2.9
6      3.0
7      3.2
8      3.2
9      3.7
10     3.9
11     4.0
12     4.0
13     4.1
14     4.5
15     4.9
16     5.1
17     5.3
18     5.9
19     6.0
20     6.8
21     7.1
22     7.9
23     8.2
24     8.7
25     9.0
26     9.5
27     9.6
28    10.3
29    10.5
Name: YearsExperience, dtype: float64
```

In [11]:
```
y
```

Out[11]:
```
0      39343
1      46205
2      37731
3      43525
4      39891
5      56642
6      60150
7      54445
8      64445
9      57189
10     63218
11     55794
12     56957
13     57081
14     61111
15     67938
16     66029
17     83088
18     81363
19     93940
20     91738
21     98273
22    101302
23    113812
24    109431
25    105582
26    116969
27    112635
28    122391
29    121872
Name: Salary, dtype: int64
```

Visualization

In [13]:
```
plt.scatter(X, y, color = 'purple')
plt.xlabel('Years of Experience');
plt.ylabel('Salary');
```



Problem Formulation

y depends on X, linearly

The equation can be y = aX + b

Splitting data: Train and Test

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 100)
```

In [17]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[17]:
```
((24,), (6,), (24,), (6,))
```

In [18]:
```
X_train
```

Out[18]:
```
27     9.6
25     9.0
6      3.0
17     5.3
22     7.9
11     4.0
4      2.2
29    10.5
0      1.1
1      1.3
18     5.9
14     4.5
19     6.0
21     7.1
2      1.5
20     6.8
10     3.9
16     5.1
15     4.9
23     8.2
7      3.2
3      2.0
24     8.7
8      3.2
Name: YearsExperience, dtype: float64
```

In [19]:
```
X_test
```

Out[19]:
```
9      3.7
26     9.5
28    10.3
13     4.1
5      2.9
12     4.0
Name: YearsExperience, dtype: float64
```

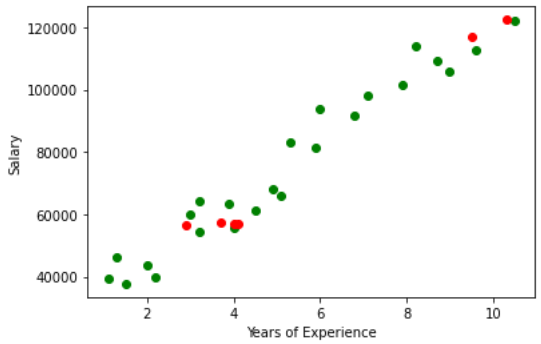
In [20]:
```
y_train
```

Out[20]:
```
27    112635
25    105582
6      60150
17     83088
22    101302
11     55794
4      39891
29    121872
0      39343
1      46205
18     81363
14     61111
19     93940
21     98273
2      37731
20     91738
10     63218
16     66029
15     67938
23    113812
7      54445
3      43525
24    109431
8      64445
Name: Salary, dtype: int64
```

In [21]:
```
y_test
```

Out[21]:
```
9      57189
26    116969
28    122391
13     57081
5      56642
12     56957
Name: Salary, dtype: int64
```

In [22]:
```
# Visualization
plt.scatter(X_train, y_train, c = 'green')
plt.scatter(X_test, y_test, c = 'red')
plt.xlabel('Years of Experience');
plt.ylabel('Salary');
```



Model Building

In [23]:
```
from sklearn.linear_model import LinearRegression
```

In [24]:
```
# Our model
lr = LinearRegression()
```

Training the model

In [25]:
```
lr.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
C:\Users\URVISH~1\AppData\Local\Temp/ipykernel_7540/4075499421.py in <module>
----> 1 lr.fit(X_train, y_train)

~\anaconda3\lib\site-packages\sklearn\linear_model\_base.py in fit(self, X, y, sample_weight)
    516         accept_sparse = False if self.positive else ['csr', 'csc', 'coo']
    517
--> 518         X, y = self._validate_data(X, y, accept_sparse=accept_sparse,
    519                                    y_numeric=True, multi_output=True)
    520

~\anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_params)
    431                 y = check_array(y, **check_y_params)
    432             else:
--> 433                 X, y = check_X_y(X, y, **check_params)
    434             out = X, y
    435

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,
copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    869         raise ValueError("y cannot be None")
    870
--> 871     X = check_array(X, accept_sparse=accept_sparse,
    872                     accept_large_sparse=accept_large_sparse,
    873                     dtype=dtype, order=order, copy=copy,

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, orde
r, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    692             # If input is 1D raise error
    693             if array.ndim == 1:
--> 694                 raise ValueError(
    695                     "Expected 2D array, got 1D array instead:\narray={}.\n"
    696                     "Reshape your data either using array.reshape(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[ 9.6  9.   3.   5.3  7.9  4.   2.2 10.5  1.1  1.3  5.9  4.5  6.   7.1
  1.5  6.8  3.9  5.1  4.9  8.2  3.2  2.   8.7  3.2].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a
single sample.
```

In [26]:
```
lr.fit(X_train.to_numpy().reshape(-1, 1), y_train)
```

Out[26]:
```
LinearRegression()
```

Prediction using the model

In [27]:
```
y_pred = lr.predict(X_test.to_numpy().reshape(-1, 1))
```

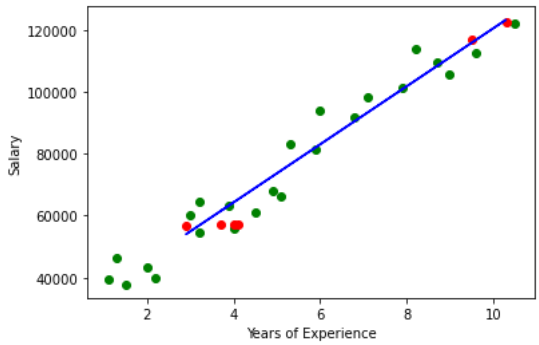
In [28]:
```
y_pred
```

Out[28]:
```
array([ 61455.19576289, 115749.67082676, 123238.56393901,  65199.64231902,
        53966.30265063,  64263.53067999])
```

In [29]:
```
y_test
```

Out[29]:
```
9      57189
26    116969
28    122391
13     57081
5      56642
12     56957
Name: Salary, dtype: int64
```

In [30]:
```
# Visualization
plt.scatter(X_train, y_train, c = 'green')
plt.scatter(X_test, y_test, c = 'red')
plt.plot(X_test, y_pred, c='blue')
plt.xlabel('Years of Experience');
plt.ylabel('Salary');
```



Regression Equation

In [31]:
```
a = lr.coef_
a
```

Out[31]:
```
array([9361.11639032])
```

In [33]:
```
b = lr.intercept_
b
```

Out[33]:
```
26819.06511870089
```

In [35]:
```
print('The regression line:', 'y =', a, 'X + ', b)
```

```
The regression line: y = [9361.11639032] X +  26819.06511870089
```

In [ ]:
```

```