Basic of ML Supervised Pridictions

Importing the libraries

In [ ]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Loading the datasets

In [ ]:

```
df = pd.read_csv("Salary_Data.csv")
```

In [ ]:

```
df
```

Out[ ]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [ ]:

```
df.shape
```

Out[ ]:

```
(30, 2)
```

```
In [ ]: df.columns
```

```
Out[ ]: Index(['YearsExperience', 'Salary'], dtype='object')
```

Check null values in datasets

```
In [ ]: df.isnull().sum()
```

```
Out[ ]: YearsExperience    0
        Salary             0
        dtype: int64
```

Checking the datatypes of datasets

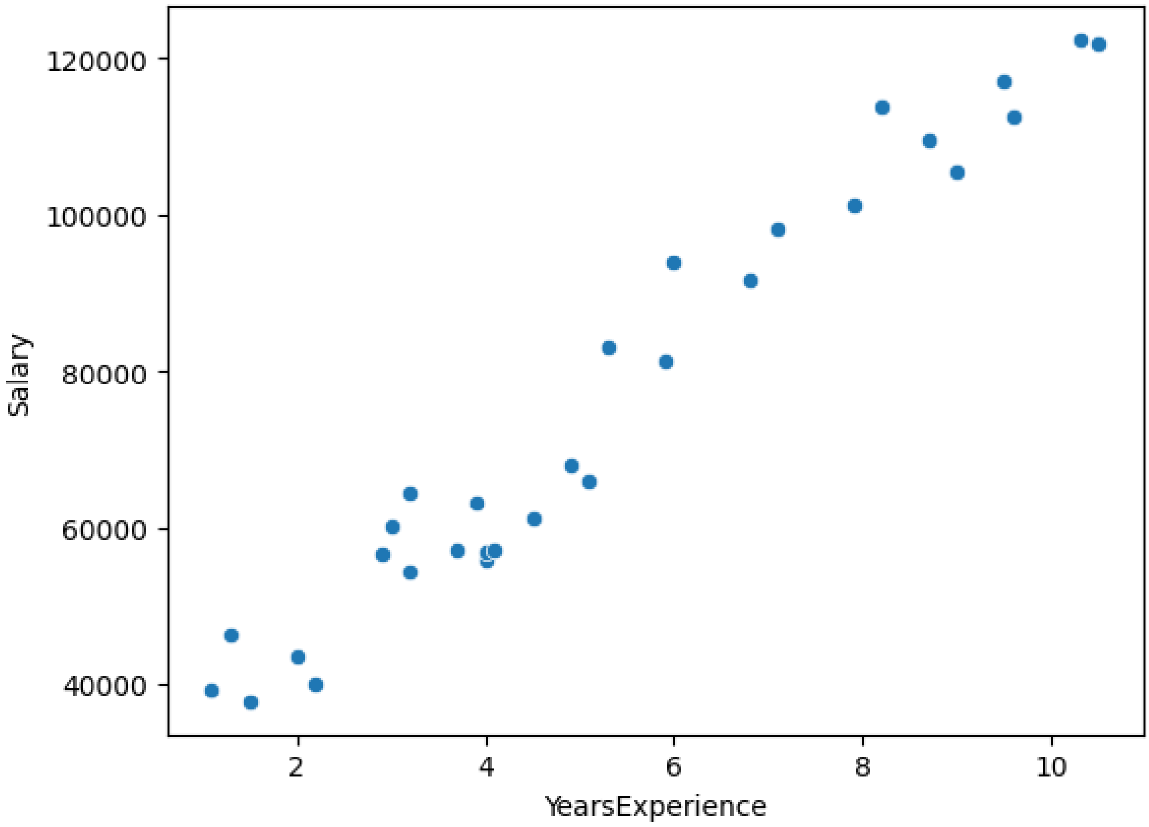
```
In [ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```

Checking the datasets is it predictable datasets or not

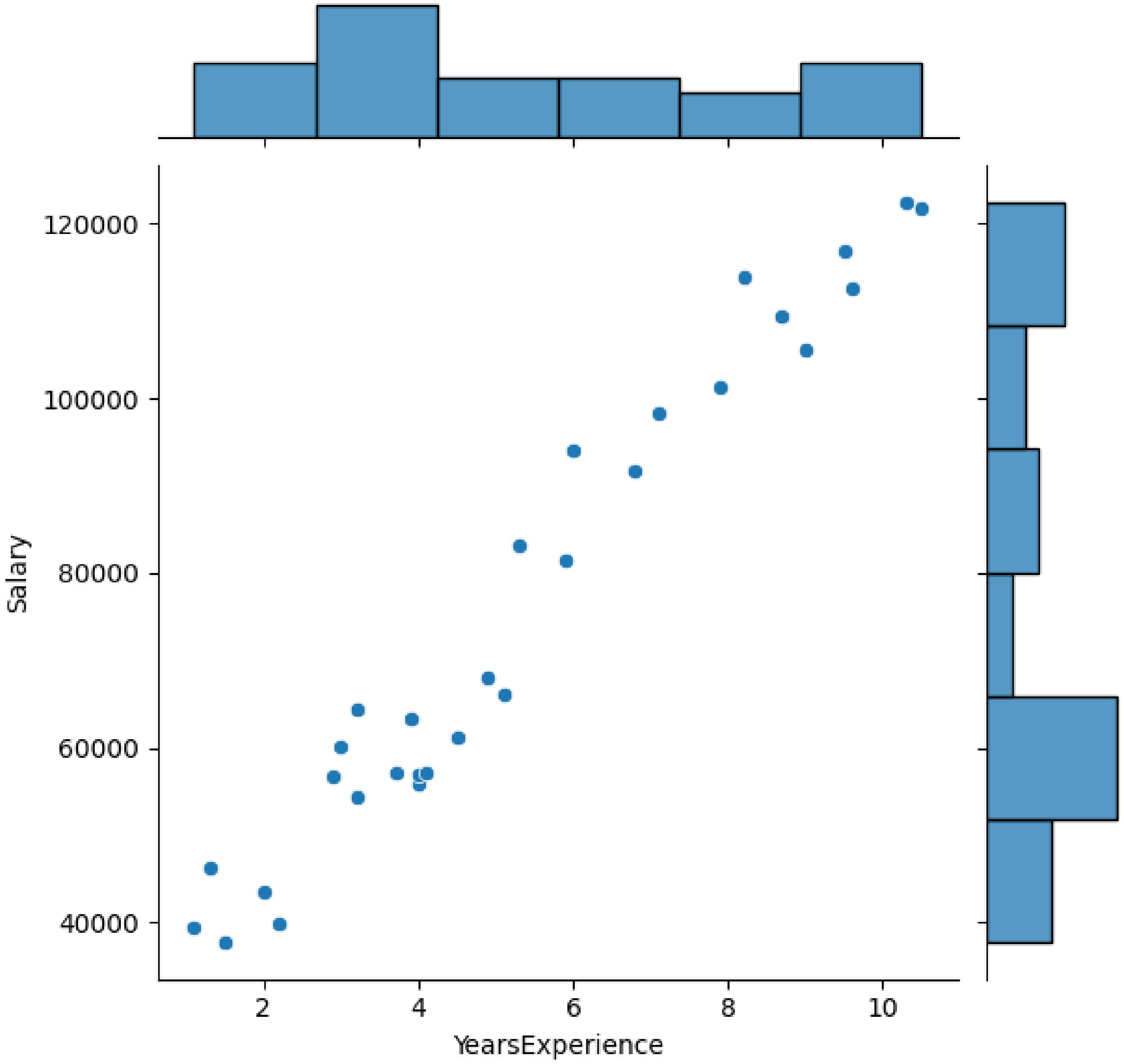
```
In [ ]: sns.scatterplot(data=df, x="YearsExperience",y="Salary")
```

```
Out[ ]: <Axes: xlabel='YearsExperience', ylabel='Salary'>
```



```
In [ ]: sns.jointplot(data=df, x="YearsExperience",y="Salary")

Out[ ]: <seaborn.axisgrid.JointGrid at 0x1ea89703210>
```



- this is the pridictable datasets

Data Preparation

- Split the data into training and testing sets:

In [ ]:

```
x = df["YearsExperience"].values.reshape(-1, 1)
y=df["Salary"]
```

In [ ]:

```
x
```

```
Out[ ]: array([[ 1.1],
               [ 1.3],
               [ 1.5],
               [ 2. ],
               [ 2.2],
               [ 2.9],
               [ 3. ],
               [ 3.2],
               [ 3.2],
               [ 3.7],
               [ 3.9],
               [ 4. ],
               [ 4. ],
               [ 4.1],
               [ 4.5],
               [ 4.9],
               [ 5.1],
               [ 5.3],
               [ 5.9],
               [ 6. ],
               [ 6.8],
               [ 7.1],
               [ 7.9],
               [ 8.2],
               [ 8.7],
               [ 9. ],
               [ 9.5],
               [ 9.6],
               [10.3],
               [10.5]])
```

```
In [ ]: y
```

Out[ ]:

```
0      39343.0
1      46205.0
2      37731.0
3      43525.0
4      39891.0
5      56642.0
6      60150.0
7      54445.0
8      64445.0
9      57189.0
10     63218.0
11     55794.0
12     56957.0
13     57081.0
14     61111.0
15     67938.0
16     66029.0
17     83088.0
18     81363.0
19     93940.0
20     91738.0
21     98273.0
22    101302.0
23    113812.0
24    109431.0
25    105582.0
26    116969.0
27    112635.0
28    122391.0
29    121872.0
Name: Salary, dtype: float64
```

In [ ]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_sta
```

In [ ]:

```
x_train
```

```
Out[ ]: array([[ 9.6],
               [ 4. ],
               [ 5.3],
               [ 7.9],
               [ 2.9],
               [ 5.1],
               [ 3.2],
               [ 4.5],
               [ 8.2],
               [ 6.8],
               [ 1.3],
               [10.5],
               [ 3. ],
               [ 2.2],
               [ 5.9],
               [ 6. ],
               [ 3.7],
               [ 3.2],
               [ 9. ],
               [ 2. ],
               [ 1.1],
               [ 7.1],
               [ 4.9],
               [ 4. ]])
```

```
In [ ]: x_test
```

```
Out[ ]: array([[ 1.5],
               [10.3],
               [ 4.1],
               [ 3.9],
               [ 9.5],
               [ 8.7]])
```

```
In [ ]: y_train
```

```
Out[ ]: 27    112635.0
        11     55794.0
        17     83088.0
        22    101302.0
        5      56642.0
        16     66029.0
        8      64445.0
        14     61111.0
        23    113812.0
        20     91738.0
        1      46205.0
        29    121872.0
        6      60150.0
        4      39891.0
        18     81363.0
        19     93940.0
        9      57189.0
        7      54445.0
        25    105582.0
        3      43525.0
        0      39343.0
        21     98273.0
        15     67938.0
        12     56957.0
        Name: Salary, dtype: float64
```

```
In [ ]: y_test
```

```
Out[ ]: 2      37731.0
        28    122391.0
        13     57081.0
        10     63218.0
        26    116969.0
        24    109431.0
        Name: Salary, dtype: float64
```

Train the Linear Regression Model

- Create and train a simple linear regression model:

```
In [ ]: model = LinearRegression()
        model.fit(x_train,y_train)
```

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

Make Predictions

- Use the trained model to make predictions on the test data and also predict the Salary for a Employees who have 8 Years of work experience:

In [ ]:

```
# Predict scores on the test data
y_predict = model.predict(x_test)
y_predict
```

Out[ ]:

```
array([ 40748.96184072, 122699.62295594,  64961.65717022,  63099.14214487,
       115249.56285456, 107799.50275317])
```

In [ ]:

```
import numpy as np

# Predict the salary for a employee who work for 8 Years
#inputs = int(input("Please input the work experience:: "))
salary_to_predict = np.array([[8]])
predicted_salary = model.predict(salary_to_predict)
predicted_salary
```

Out[ ]:

```
array([101280.70016446])
```

In [ ]:

```
# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_predict)
mse = mean_squared_error(y_test, y_predict)
r2 = r2_score(y_test, y_predict)
```

In [ ]:

```
mae
```

Out[ ]:

```
2446.1723690465055
```

In [ ]:

```
mse
```

Out[ ]:

```
12823412.298126549
```

In [ ]:

```
r2
```

Out[ ]:

```
0.988169515729126
```