In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
```

In [2]:

```
dataset=pd.read_csv('Salary_Data.csv')
dataset.head()
```

Out[2]:

	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

In [3]:

```
#Null value and data types check
dataset.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: 608.0 bytes
```

In [4]:

```
#rename the Delivery Time column as delivery_time and Sorting Time Column as sorting_time
Salary1= dataset.rename({'YearsExperience': 'YrExp'}, axis=1)
Salary1.head()
```

Out[4]:

	YrExp	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

In [5]:

```
#Print the duplicated rows if present inside the data set
Salary1[dataset.duplicated(keep = False)]
```

Out[5]:

YrExp	Salary

Hence as per above process we found that there is no duplicate values are present inside the data set

In [6]:

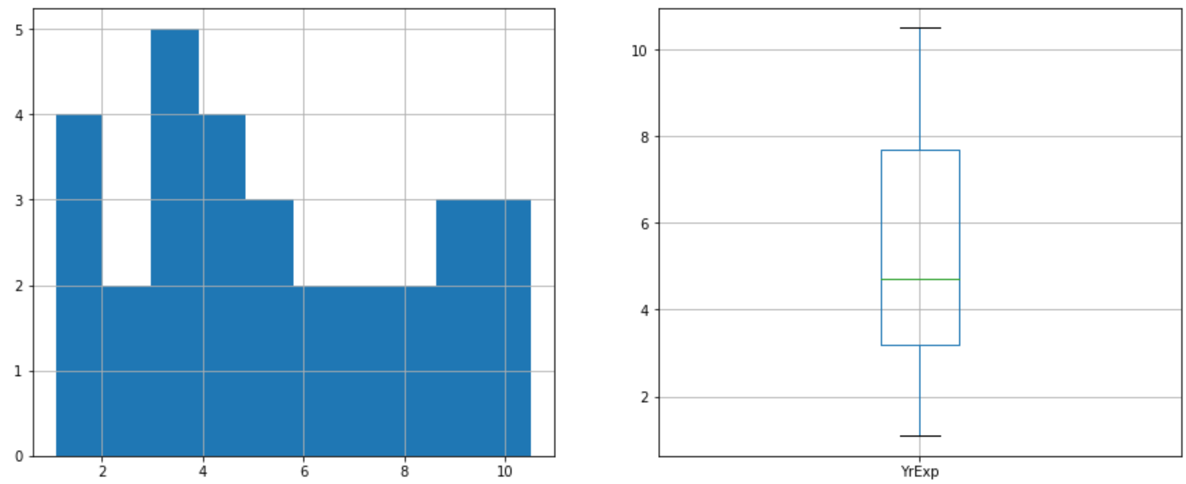
```
#Correlation
dataset.corr()
```

Out[6]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

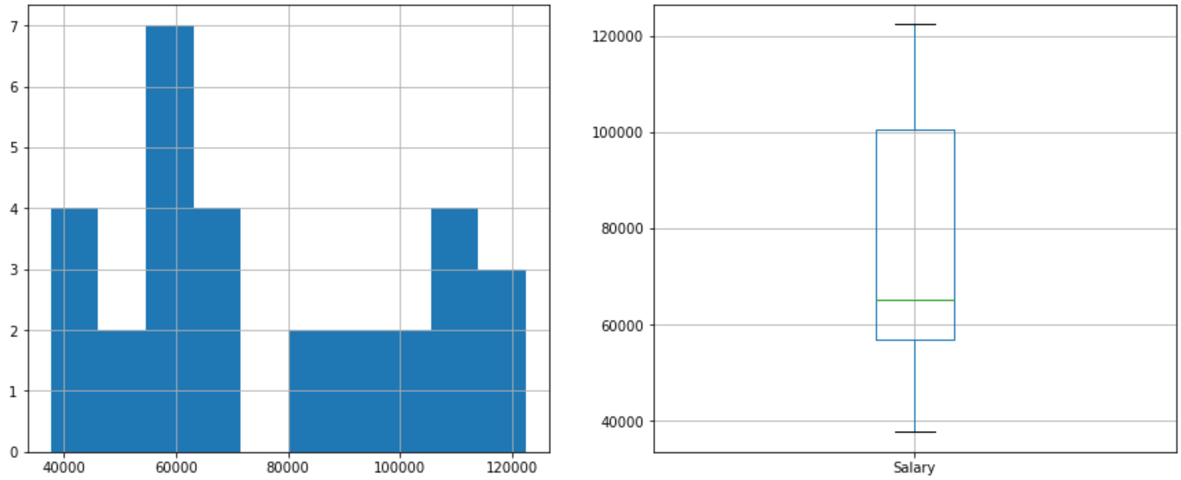
In [36]:

```
#Outlier checking-checking whether outliers are present in YrExp column
plt.figure(figsize = (15,6))
plt.subplot(1,2,1)
Salary1['YrExp'].hist()
plt.subplot(1,2,2)
Salary1.boxplot(column=['YrExp'])

plt.show()
```



In [37]:

```
plt.figure(figsize = (15,6))
plt.subplot(1,2,1)
Salary1['Salary'].hist()
plt.subplot(1,2,2)
Salary1.boxplot(column=['Salary'])

plt.show()
```



From the above plots, we found that there is no outleirs present inside the YrExp and Salary data column

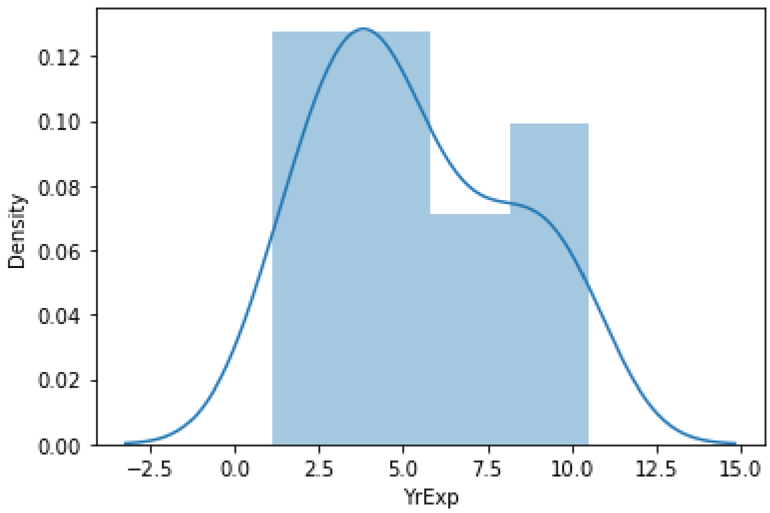
In [8]:

```
#Cheking of distribution of data using distplot
sns.distplot(Salary1['YrExp'])
```

```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\distributions.p
y:2619: FutureWarning: `distplot` is a deprecated function and will be remov
ed in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-level f
unction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

```
<AxesSubplot:xlabel='YrExp', ylabel='Density'>
```



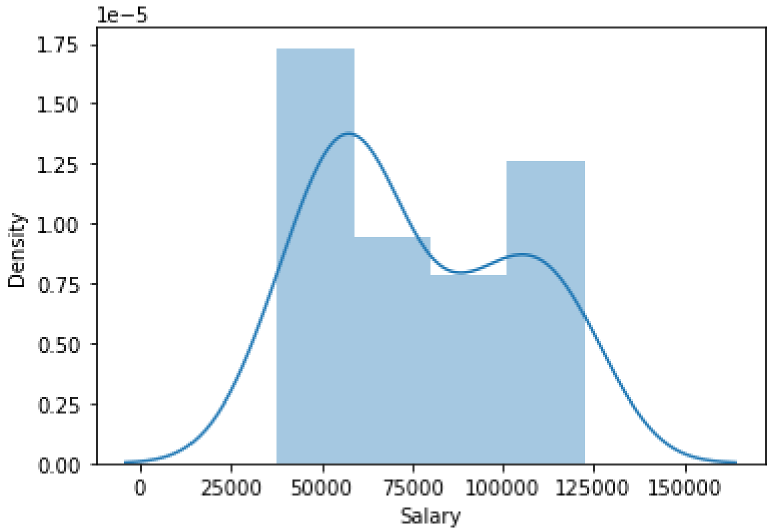
In [9]:

```
sns.distplot(Salary1['Salary'])
```

```
C:\Users\sowmya sandeep\anaconda3\lib\site-packages\seaborn\distributions.p
y:2619: FutureWarning: `distplot` is a deprecated function and will be remov
ed in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-level f
unction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[9]:

```
<AxesSubplot:xlabel='Salary', ylabel='Density'>
```



Try to fit in a model for Salary_hike

In [10]:

```
#Predict a model without applying transformation
```

In [11]:

```
dataset_1= smf.ols("Salary~YrExp",data= Salary1).fit()
```

In [12]:

```
dataset_1
```

Out[12]:

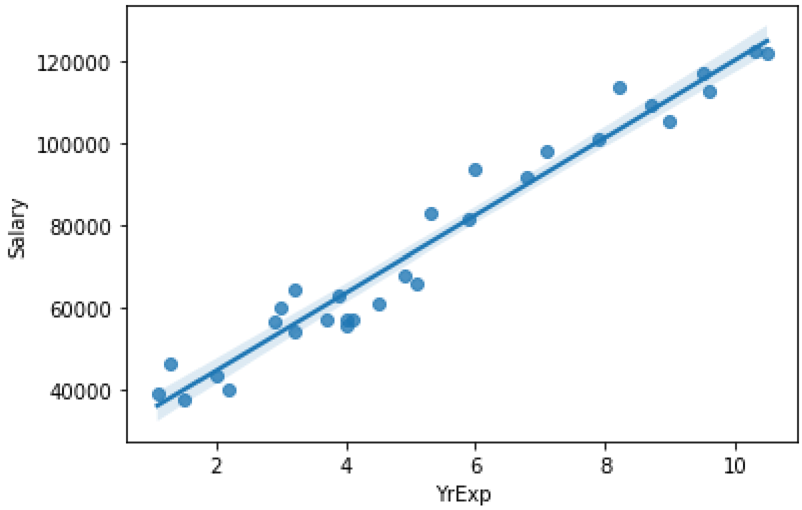
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x22918bd2a
00>
```

In [13]:

```
#Regresssion Plot
sns.regplot(x="YrExp", y="Salary", data = Salary1)
```

Out[13]:

```
&lt;AxesSubplot:xlabel='YrExp', ylabel='Salary'&gt;
```



In [14]:

```
#Coefficients
dataset_1.params
```

Out[14]:

```
Intercept    25792.200199
YrExp         9449.962321
dtype: float64
```

In [15]:

```
print(dataset_1.tvalues, '\n', dataset_1.pvalues)
```

```
Intercept    11.346940
YrExp        24.950094
dtype: float64
 Intercept    5.511950e-12
YrExp        1.143068e-20
dtype: float64
```

In [16]:

```
(dataset_1.rsquared,dataset_1.rsquared_adj)
```

Out[16]:

```
(0.9569566641435086, 0.9554194021486339)
```

In [17]:

```
dataset_1.summary()
```

Out[17]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.957
Model:	OLS	**Adj. R-squared:**	0.955
Method:	Least Squares	**F-statistic:**	622.5
Date:	Wed, 25 May 2022	**Prob (F-statistic):**	1.14e-20
Time:	16:33:17	**Log-Likelihood:**	-301.44
No. Observations:	30	**AIC:**	606.9
Df Residuals:	28	**BIC:**	609.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	2.579e+04	2273.053	11.347	0.000	2.11e+04	3.04e+04
YrExp	9449.9623	378.755	24.950	0.000	8674.119	1.02e+04

Omnibus:	2.140	**Durbin-Watson:**	1.648
Prob(Omnibus):	0.343	**Jarque-Bera (JB):**	1.569
Skew:	0.363	**Prob(JB):**	0.456
Kurtosis:	2.147	**Cond. No.**	13.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

As per above OLS Regression Results dependent variable is salary and here p value is less than 0.05 hence this is significant model Here R-squred value is 0.957, which is greater than 0.8. Hence we can say our model is good for Salary_hike

In [18]:

```
def RMSE(predict, actual):
    return np.sqrt(((predict - actual) ** 2).mean())
```

In [20]:

```
#Checking the RMSE value
value_1 = dataset_1.predict(Salary1.YrExp)
value_1
```

Out[20]:

```
0      36187.158752
1      38077.151217
2      39967.143681
3      44692.124842
4      46582.117306
5      53197.090931
6      54142.087163
7      56032.079627
8      56032.079627
9      60757.060788
10     62647.053252
11     63592.049484
12     63592.049484
13     64537.045717
14     68317.030645
15     72097.015574
16     73987.008038
17     75877.000502
18     81546.977895
19     82491.974127
20     90051.943985
21     92886.932681
22    100446.902538
23    103281.891235
24    108006.872395
25    110841.861092
26    115566.842252
27    116511.838485
28    123126.812110
29    125016.804574
dtype: float64
```

In [23]:

```
actual = Salary1.Salary
```

In [25]:

```
RMSE(value_1,actual)
```

Out[25]:

```
5592.043608760662
```

Model1 - We may apply transformation on variables to get better R-squared value as to predict better model

In [26]:

```
#Applying Logarithmic Transformation and Predict a new model
```

In [28]:

```
dataset_2 = smf.ols("Salary~np.log(YrExp)",data = Salary1).fit()
```

In [29]:

```
dataset_2.summary()
```

Out[29]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.854
Model:	OLS	**Adj. R-squared:**	0.849
Method:	Least Squares	**F-statistic:**	163.6
Date:	Wed, 25 May 2022	**Prob (F-statistic):**	3.25e-13
Time:	16:36:41	**Log-Likelihood:**	-319.77
No. Observations:	30	**AIC:**	643.5
Df Residuals:	28	**BIC:**	646.3
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.493e+04	5156.226	2.895	0.007	4365.921	2.55e+04
np.log(YrExp)	4.058e+04	3172.453	12.792	0.000	3.41e+04	4.71e+04

Omnibus:	1.094	**Durbin-Watson:**	0.512
Prob(Omnibus):	0.579	**Jarque-Bera (JB):**	0.908
Skew:	0.156	**Prob(JB):**	0.635
Kurtosis:	2.207	**Cond. No.**	5.76

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [33]:

```
#Checking of RMSE value of the new model1
value_2 = dataset_2.predict(Salary1.YrExp)
value_2
```

Out[33]:

```
0      18795.848339
1      25575.235192
2      31382.551905
3      43057.262306
4      46925.138875
5      58136.050079
6      59511.842441
7      62130.943929
8      62130.943929
9      68022.718504
10     70159.105863
11     71186.552842
12     71186.552842
13     72188.628149
14     75966.422577
15     79422.295729
16     81045.791737
17     82606.829882
18     86959.066704
19     87641.132977
20     92720.502137
21     94472.514696
22     98805.371390
23    100317.918684
24    102719.920751
25    104095.713112
26    106289.868435
27    106714.814600
28    109571.007247
29    110351.454145
dtype: float64
```

In [34]:

```
RMSE(value_2,actual)
```

Out[34]:

```
10302.893706228302
```

#Here after applying logarithmic transformation on YrExp varibale, from Model1 we get that R-squared value is 0.854 and p value is less than 0.05. Conclusion - Comparing between model and model1 , model has higher R-squared value i.e. 0.957 as comapare to model1. And also RMSE value is lower in model as compare to model1. From the above data we know higher R-squred value and lower RMSE value gives better model. Hence the first model i.e. model is better model to predict Salary_hike

In [ ]:

```

```