In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

```

In [2]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
salary_data=pd.read_csv('Salary_Data.csv')
salary_data
```

Out[3]:

	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

	YearsExperience	Salary
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 [4]:
```
salary_data.isnull().sum()
```

Out[4]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [5]:
```
salary_data.dtypes
```

Out[5]:
```
YearsExperience    float64
Salary             float64
dtype: object
```

In [6]:
```
salary_data.shape
```

Out[6]:
```
(30, 2)
```

In [7]:
```
salary_data.info()
```

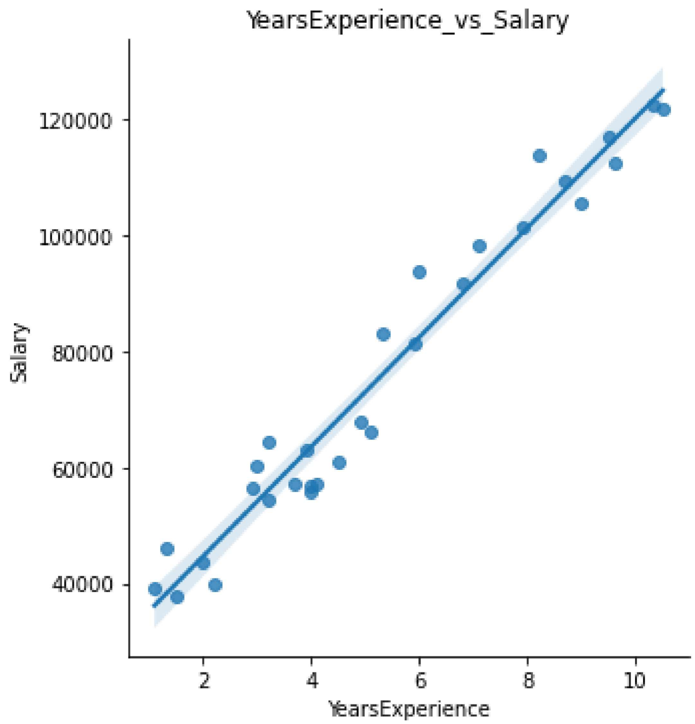
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Assumption check

1. linearity check

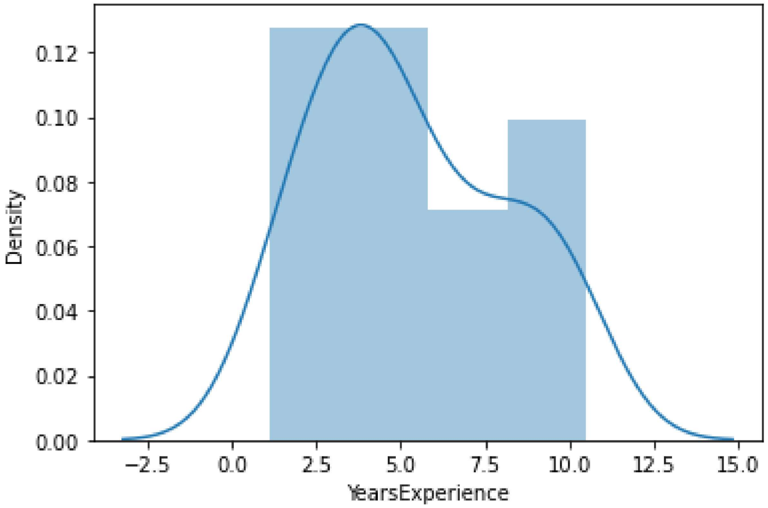
In [8]:

```
sns.lmplot(x='YearsExperience',y='Salary',data=salary_data)
plt.title('YearsExperience_vs_Salary')
plt.show()
```



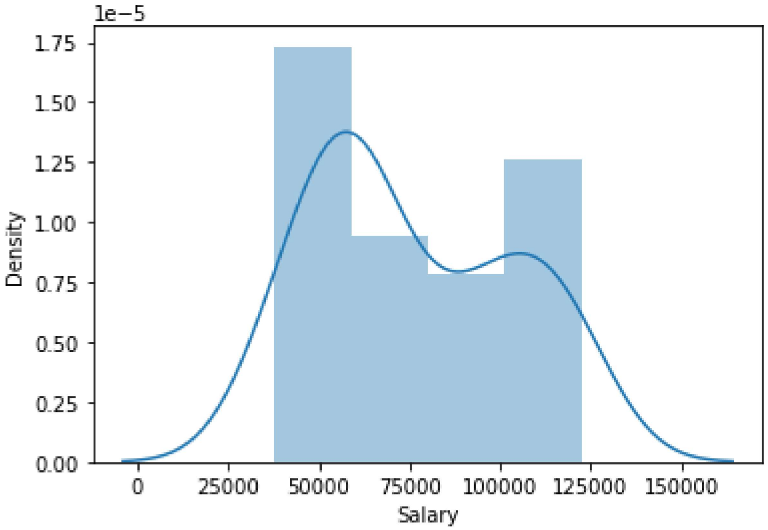
2. distribution check

In [9]:
```
sns.distplot(a=salary_data['YearsExperience'],hist=True)
plt.show()
```



In [10]:

```
sns.distplot(a=salary_data['Salary'],hist=True)
plt.show()
```



In [11]:

```
salary_data.corr()
```

Out[11]:

	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [12]:
```
sns.heatmap(salary_data.corr(),annot=True)
plt.show()
```

model building || model training

In [13]:
```
import statsmodels.formula.api as smf
```

In [14]:
```
linear_model=smf.ols(formula='Salary~YearsExperience',data=salary_data).fit()
linear_model
```

Out[14]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1a0255e0f70>
```

In [15]:
```
linear_model.params
```

Out[15]:
```
Intercept          25792.200199
YearsExperience     9449.962321
dtype: float64
```

In [16]:
```
linear_model.pvalues
```

Out[16]:
```
Intercept          5.511950e-12
YearsExperience    1.143068e-20
dtype: float64
```

model testing

In [17]:
```
y=(3*9449.962321)+ 25792.200199 #manual prediction for 3 years of experience
y
```

Out[17]:
```
54142.087162
```

In [18]:
```
z=(4*9449.962321)+ 25792.200199 #manual prediction for 4 years of experience
z
```

Out[18]:
```
63592.049483
```

In [19]:
```
data=pd.Series([3,4])# auto prediction for 3 and 4 years of experience
data
```

Out[19]:
```
0    3
1    4
dtype: int64
```

In [20]:
```
data_pred=pd.DataFrame(data,columns=['YearsExperience'])
data_pred
```

Out[20]:

	YearsExperience
0	3
1	4

In [21]:
```
model_pred=linear_model.predict(data_pred)
model_pred
```

Out[21]:
```
0    54142.087163
1    63592.049484
dtype: float64
```

model evaluation

In [22]:
```
linear_model=smf.ols(formula='Salary~YearsExperience',data=salary_data).fit()
print('R-square                           : ',round(linear_model.rsquared,4))
print('Adjusted R-square                  : ',round(linear_model.rsquared_adj,4))
print('Akaike information criterion (AIC) : ',round(linear_model.aic,4))
print('Bayesian information criterion(BIC): ',round(linear_model.bic,4))
```

```
R-square                           :  0.957
Adjusted R-square                  :  0.9554
Akaike information criterion (AIC) :  606.8823
Bayesian information criterion(BIC):  609.6847
```

In [37]:
```
plt.figure(figsize = (8,6))
sns.scatter(x = linear_model["YearsExperience"], y = linear_model["Salary"],data=salary_data)
plt.title("Hetroscedasticity", fontweight = 'bold', fontsize = 14)
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_752/4037791447.py in <module>
      1 plt.figure(figsize = (8,6))
----> 2 sns.scatter(x = linear_model["YearsExperience"], y = linear_model["Salary"],data=salary_data)
      3 plt.title("Hetroscedasticity", fontweight = 'bold', fontsize = 14)
      4 plt.show()

AttributeError: module 'seaborn' has no attribute 'scatter'
```

```
<Figure size 576x432 with 0 Axes>
```

1.Log Transformation

In [23]:
```
salary_data['log_YearsExperience']=np.log(salary_data['YearsExperience'])
salary_data.head(10)
```

Out[23]:

	YearsExperience	Salary	log_YearsExperience
0	1.1	39343.0	0.095310
1	1.3	46205.0	0.262364
2	1.5	37731.0	0.405465
3	2.0	43525.0	0.693147
4	2.2	39891.0	0.788457
5	2.9	56642.0	1.064711
6	3.0	60150.0	1.098612
7	3.2	54445.0	1.163151
8	3.2	64445.0	1.163151
9	3.7	57189.0	1.308333

In [24]:
```
sns.distplot(a=salary_data['log_YearsExperience'])
plt.title('log_YearsExperience')
plt.show()
```

In [25]:
```
model_1=smf.ols(formula='Salary~log_YearsExperience', data=salary_data).fit()
model_1.summary()
```

Out[25]:

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:	Fri, 01 Jul 2022	**Prob (F-statistic):**	3.25e-13
Time:	12:09:50	**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
log_YearsExperience	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 [26]:
```
model_1.rsquared
```

Out[26]: 0.8538888828756969

This r-square value is less than the r-square of the model from raw data.

2. SQRT MODEL

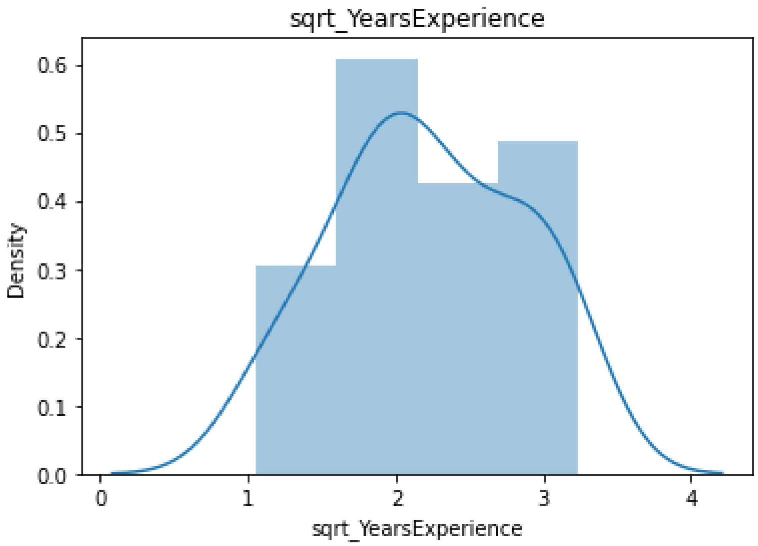
In [27]:
```
salary_data['sqrt_YearsExperience']=np.sqrt(salary_data['YearsExperience'])
salary_data.head(10)
```

Out[27]:

	YearsExperience	Salary	log_YearsExperience	sqrt_YearsExperience
0	1.1	39343.0	0.095310	1.048809
1	1.3	46205.0	0.262364	1.140175
2	1.5	37731.0	0.405465	1.224745
3	2.0	43525.0	0.693147	1.414214
4	2.2	39891.0	0.788457	1.483240
5	2.9	56642.0	1.064711	1.702939
6	3.0	60150.0	1.098612	1.732051
7	3.2	54445.0	1.163151	1.788854
8	3.2	64445.0	1.163151	1.788854
9	3.7	57189.0	1.308333	1.923538

In [28]:

```
sns.distplot(a=salary_data['sqrt_YearsExperience'])
plt.title('sqrt_YearsExperience')
plt.show()
```



In [29]:
```
model_2=smf.ols(formula='Salary~sqrt_YearsExperience', data=salary_data).fit()
model_2.summary()
```

Out[29]:

OLS Regression Results

Dep. Variable:	Salary	**R-squared:**	0.931
Model:	OLS	**Adj. R-squared:**	0.929
Method:	Least Squares	**F-statistic:**	377.8
Date:	Fri, 01 Jul 2022	**Prob (F-statistic):**	8.57e-18
Time:	12:09:50	**Log-Likelihood:**	-308.52
No. Observations:	30	**AIC:**	621.0
Df Residuals:	28	**BIC:**	623.8
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	-1.606e+04	4921.599	-3.262	0.003	-2.61e+04	-5974.331
sqrt_YearsExperience	4.15e+04	2135.122	19.437	0.000	3.71e+04	4.59e+04

Omnibus:	0.588	**Durbin-Watson:**	1.031
Prob(Omnibus):	0.745	**Jarque-Bera (JB):**	0.638
Skew:	0.011	**Prob(JB):**	0.727
Kurtosis:	2.286	**Cond. No.**	9.97

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

In [30]:
```
model_2.rsquared
```

Out[30]: 0.9310009544993526

This r-square value is less than the r-square of the model from raw data.

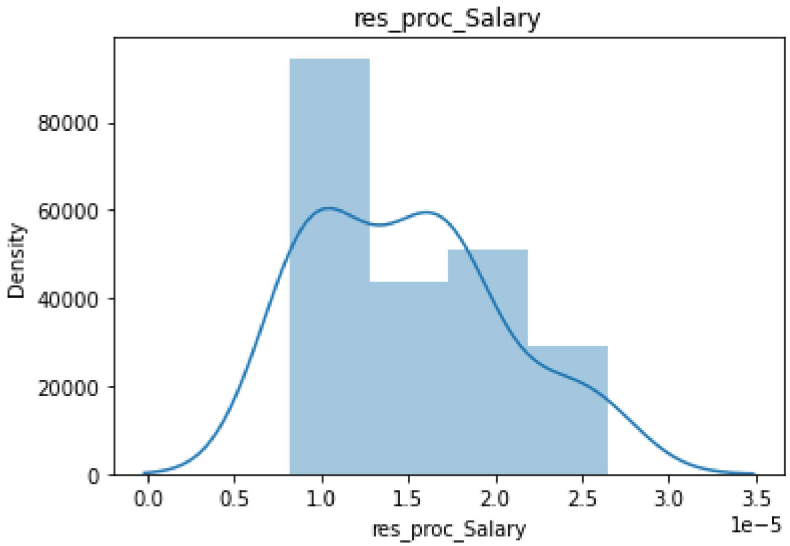
3. RECIPROCAL

In [31]:
```
salary_data['res_proc_Salary']=1/salary_data['Salary']
salary_data.head(10)
```

Out[31]:

	YearsExperience	Salary	log_YearsExperience	sqrt_YearsExperience	res_proc_Salary
0	1.1	39343.0	0.095310	1.048809	0.000025
1	1.3	46205.0	0.262364	1.140175	0.000022
2	1.5	37731.0	0.405465	1.224745	0.000027
3	2.0	43525.0	0.693147	1.414214	0.000023
4	2.2	39891.0	0.788457	1.483240	0.000025
5	2.9	56642.0	1.064711	1.702939	0.000018
6	3.0	60150.0	1.098612	1.732051	0.000017
7	3.2	54445.0	1.163151	1.788854	0.000018
8	3.2	64445.0	1.163151	1.788854	0.000016
9	3.7	57189.0	1.308333	1.923538	0.000017

In [32]:
```
sns.distplot(a=salary_data['res_proc_Salary'])
plt.title('res_proc_Salary')
plt.show()
```



In [33]:
```
model_3=smf.ols(formula='res_proc_Salary~YearsExperience', data=salary_data).fit()
model_3.summary()
```

Out[33]:

OLS Regression Results

Dep. Variable:	res_proc_Salary	**R-squared:**	0.861
Model:	OLS	**Adj. R-squared:**	0.856
Method:	Least Squares	**F-statistic:**	173.2
Date:	Fri, 01 Jul 2022	**Prob (F-statistic):**	1.63e-13
Time:	12:09:50	**Log-Likelihood:**	350.83
No. Observations:	30	**AIC:**	-697.7
Df Residuals:	28	**BIC:**	-694.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.454e-05	8.2e-07	29.913	0.000	2.29e-05	2.62e-05
YearsExperience	-1.799e-06	1.37e-07	-13.162	0.000	-2.08e-06	-1.52e-06

Omnibus:	1.760	**Durbin-Watson:**	1.137
Prob(Omnibus):	0.415	**Jarque-Bera (JB):**	1.380
Skew:	0.516	**Prob(JB):**	0.502
Kurtosis:	2.802	**Cond. No.**	13.2

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

In [34]:
```
model_3.rsquared
```

Out[34]: 0.8608672473082564

This r-square value is less than the r-square of the model from raw data.

<u>Model Selection</u>

Now by comparing r-square of all models,

we can say that the models which are fitted by using transformation are not so good as compare to our model from raw data(original data)

Hence , we select our first model for further calculation