Simple Linear Regression Assignment

Data Set : Salary_Data

Q = Build a Prediction Model for Salary_Hike

1. Import Necessary libraries

In [1]:
```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

import statsmodels.formula.api as smf

from sklearn.metrics import mean_squared_error
from math import sqrt

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

2. Import Data

In [2]:
```python
salary_details = pd.read_csv('Salary_Data.csv')
salary_details
```

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
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

3. Data Understanding

a) Initial Analysis:

In [3]:
```python
salary_details.head()
```

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

In [4]:
```python
salary_details.shape
```

Out[4]:
```
(30, 2)
```

In [5]:
```python
salary_details.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 [6]:
```python
salary_details.isna().sum()
```

Out[6]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [7]:
```python
salary_details.dtypes
```

Out[7]:
```
YearsExperience    float64
Salary             float64
dtype: object
```

There is no Null value present in this data set and also the data types are appropriate in all attributes

b) Correlation Matrix :

In [8]:
```python
corr_matrix = salary_details.corr()
corr_matrix
```

Out[8]:

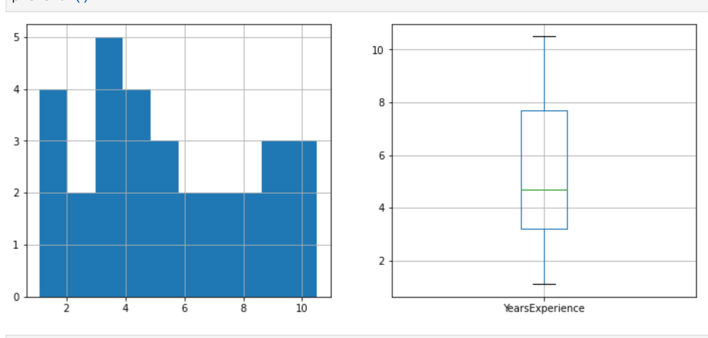
	YearsExperience	Salary
YearsExperience	1.000000	0.978242
Salary	0.978242	1.000000

In [9]:
```python
sns.heatmap(data = corr_matrix,annot = True)
plt.show()
```

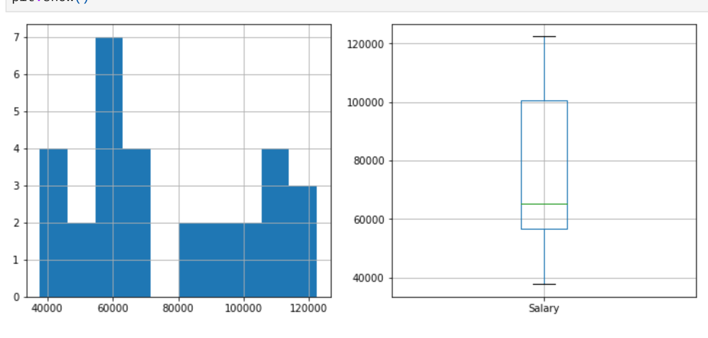
5. Perform Assumption Check

a) Outlier Test Using Box Plot :

In [10]:
```python
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
salary_details['YearsExperience'].hist()
plt.subplot(1,2,2)
salary_details.boxplot(column = ['YearsExperience'])

plt.show()
```



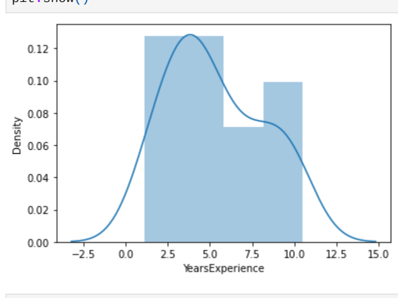
In [11]:
```python
plt.figure(figsize = (12,5))
plt.subplot(1,2,1)
salary_details['Salary'].hist()
plt.subplot(1,2,2)
salary_details.boxplot(column = ['Salary'])

plt.show()
```



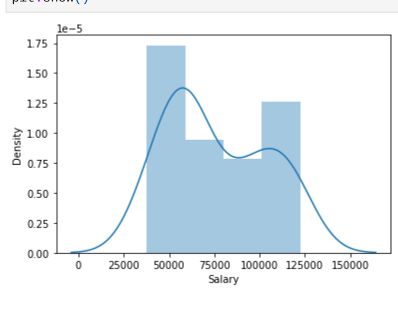
From the above histogrms and boxplots, we found that there is no outleirs present inside the YearsExperience and Salary data.

b) Normality / Distribution Test Using Distplot :

In [12]:
```python
sns.distplot(salary_details['YearsExperience'])
plt.show()
```



In [13]:
```python
sns.distplot(salary_details['Salary'])
plt.show()
```



Normality Test Failed

7. Model Building || 8. Model Training

Now Try to fit Model for Salary Hike

Model 1: Without Applying any Transformation

Using Statsmodel

In [14]:
```python
model_1 = smf.ols(formula = 'YearsExperience~Salary', data = salary_details).fit()
model_1
```

Out[14]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e16289e760>
```

In [15]:
```python
#coefficient
model_1.params
```

Out[15]:
```
Intercept   -2.383161
Salary       0.000101
dtype: float64
```

In [16]:
```python
model_1.summary()
```

Out[16]:

OLS Regression Results

Dep. Variable:	YearsExperience	R-squared:	0.957
Model:	OLS	Adj. R-squared:	0.955
Method:	Least Squares	F-statistic:	622.5
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	1.14e-20
Time:	00:42:15	Log-Likelihood:	-26.168
No. Observations:	30	AIC:	56.34
Df Residuals:	28	BIC:	59.14
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.3832	0.327	-7.281	0.000	-3.054	-1.713
Salary	0.0001	4.06e-06	24.950	0.000	9.3e-05	0.000

Omnibus:	3.544	Durbin-Watson:	1.587
Prob(Omnibus):	0.170	Jarque-Bera (JB):	2.094
Skew:	-0.412	Prob(JB):	0.351
Kurtosis:	2.003	Cond. No.	2.41e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.41e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

From the Above OLS Regression Result the R-Squared value is 0.957 > 0.75 and we can say that this Model is good to Predict Salary_hike and p-value < 0.05 and it is significant model

Model 2: Apply Log Transformation of Y

In [17]:
```python
model_2 = smf.ols(formula = 'Salary~np.log(YearsExperience)',data = salary_details).fit()
model_2
```

Out[17]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168f997c0>
```

In [18]:
```python
model_2.params
```

Out[18]:
```
Intercept                  14927.97177
np.log(YearsExperience)    40581.98796
dtype: float64
```

In [19]:
```python
model_2.summary()
```

Out[19]:

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:	Sat, 24 Sep 2022	Prob (F-statistic):	3.25e-13
Time:	00:42:17	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(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.

Model 3: Apply Log Transformation of X

In [20]:
```python
model_3 = smf.ols(formula = 'np.log(Salary)~YearsExperience',data = salary_details).fit()
model_3
```

Out[20]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168f6f580>
```

In [21]:
```python
model_3.params
```

Out[21]:
```
Intercept          10.507402
YearsExperience     0.125453
dtype: float64
```

In [22]:
```python
model_3.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	np.log(Salary)	R-squared:	0.932
Model:	OLS	Adj. R-squared:	0.930
Method:	Least Squares	F-statistic:	383.6
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	7.03e-18
Time:	00:42:18	Log-Likelihood:	28.183
No. Observations:	30	AIC:	-52.37
Df Residuals:	28	BIC:	-49.56
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	10.5074	0.038	273.327	0.000	10.429	10.586
YearsExperience	0.1255	0.006	19.585	0.000	0.112	0.139

Omnibus:	0.826	Durbin-Watson:	1.438
Prob(Omnibus):	0.661	Jarque-Bera (JB):	0.812
Skew:	0.187	Prob(JB):	0.666
Kurtosis:	2.286	Cond. No.	13.2

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

Model 4: Apply Log Transformation of X and Y

In [23]:
```python
model_4 = smf.ols(formula = 'np.log(Salary)~np.log(YearsExperience)',data = salary_details).fit()
model_4
```

Out[23]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168fa49d0>
```

In [24]:
```python
model_4.params
```

Out[24]:
```
Intercept                  10.328043
np.log(YearsExperience)     0.562089
dtype: float64
```

In [25]:
```python
model_4.summary()
```

Out[25]:

OLS Regression Results

Dep. Variable:	np.log(Salary)	R-squared:	0.905
Model:	OLS	Adj. R-squared:	0.902
Method:	Least Squares	F-statistic:	267.4
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	7.40e-16
Time:	00:42:19	Log-Likelihood:	23.209
No. Observations:	30	AIC:	-42.42
Df Residuals:	28	BIC:	-39.61
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	10.3280	0.056	184.868	0.000	10.214	10.442
np.log(YearsExperience)	0.5621	0.034	16.353	0.000	0.492	0.632

Omnibus:	0.102	Durbin-Watson:	0.988
Prob(Omnibus):	0.950	Jarque-Bera (JB):	0.297
Skew:	0.093	Prob(JB):	0.862
Kurtosis:	2.549	Cond. No.	5.76

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

Model 5: Apply Exponential Transformation

In [26]:
```python
model_5 = smf.ols(formula = 'Salary~np.exp(YearsExperience)',data = salary_details).fit()
model_5
```

Out[26]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168f99af0>
```

In [27]:
```python
model_5.params
```

Out[27]:
```
Intercept                  67568.624969
np.exp(YearsExperience)        2.136040
dtype: float64
```

In [28]:
```python
model_5.summary()
```

Out[28]:

OLS Regression Results

Dep. Variable:	Salary	R-squared:	0.472
Model:	OLS	Adj. R-squared:	0.454
Method:	Least Squares	F-statistic:	25.07
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	2.72e-05
Time:	00:42:21	Log-Likelihood:	-339.03
No. Observations:	30	AIC:	682.1
Df Residuals:	28	BIC:	684.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.757e+04	4065.396	16.620	0.000	5.92e+04	7.59e+04
np.exp(YearsExperience)	2.1360	0.427	5.007	0.000	1.262	3.010

Omnibus:	4.567	Durbin-Watson:	0.202
Prob(Omnibus):	0.102	Jarque-Bera (JB):	1.966
Skew:	0.276	Prob(JB):	0.374
Kurtosis:	1.874	Cond. No.	1.05e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.05e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Model 6: Apply Reciprocal Transformation

In [29]:
```python
model_6 = smf.ols(formula = 'Salary~np.reciprocal(YearsExperience)',data = salary_details).fit()
model_6
```

Out[29]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168f39c10>
```

In [30]:
```python
model_6.params
```

Out[30]:
```
Intercept                         104273.335111
np.reciprocal(YearsExperience)   -103620.843905
dtype: float64
```

In [31]:
```python
model_6.summary()
```

Out[31]:

OLS Regression Results

Dep. Variable:	Salary	R-squared:	0.589
Model:	OLS	Adj. R-squared:	0.574
Method:	Least Squares	F-statistic:	40.06
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	7.58e-07
Time:	00:42:22	Log-Likelihood:	-335.30
No. Observations:	30	AIC:	674.6
Df Residuals:	28	BIC:	677.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.043e+05	5533.996	18.842	0.000	9.29e+04	1.16e+05
np.reciprocal(YearsExperience)	-1.036e+05	1.64e+04	-6.329	0.000	-1.37e+05	-7.01e+04

Omnibus:	10.284	Durbin-Watson:	0.220
Prob(Omnibus):	0.006	Jarque-Bera (JB):	2.740
Skew:	0.290	Prob(JB):	0.254
Kurtosis:	1.638	Cond. No.	5.40

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

Model 7: Apply Square Transformation

In [32]:
```python
model_7 = smf.ols(formula = 'Salary~np.square(YearsExperience)',data = salary_details).fit()
model_7
```

Out[32]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168f39760>
```

In [33]:
```python
model_7.params
```

Out[33]:
```
Intercept                     48042.585515
np.square(YearsExperience)      776.318773
dtype: float64
```

In [34]:
```python
model_7.summary()
```

Out[34]:

OLS Regression Results

Dep. Variable:	Salary	R-squared:	0.915
Model:	OLS	Adj. R-squared:	0.912
Method:	Least Squares	F-statistic:	302.7
Date:	Sat, 24 Sep 2022	Prob (F-statistic):	1.52e-16
Time:	00:42:23	Log-Likelihood:	-311.59
No. Observations:	30	AIC:	627.2
Df Residuals:	28	BIC:	630.0
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	4.804e+04	2186.372	21.974	0.000	4.36e+04	5.25e+04
np.square(YearsExperience)	776.3188	44.624	17.397	0.000	684.911	867.726

Omnibus:	1.294	Durbin-Watson:	0.883
Prob(Omnibus):	0.524	Jarque-Bera (JB):	1.240
Skew:	0.409	Prob(JB):	0.538
Kurtosis:	2.432	Cond. No.	72.3

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

Model 8: Apply Square Root Transformation

In [35]:
```python
model_8 = smf.ols(formula = 'Salary~np.sqrt(YearsExperience)',data = salary_details).fit()
model_8
```

Out[35]:
```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1e168a21a90>
```

In [36]:
```python
model_8.params
```

Out[36]:
```
Intercept                  -16055.769117
np.sqrt(YearsExperience)    41500.680583
dtype: float64
```

In [37]:
```python
model_8.summary()
```

Out[37]:

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:	Sat, 24 Sep 2022	Prob (F-statistic):	8.57e-18
Time:	00:42:24	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
np.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.

CONCLUSION = Comparing between all Models we got to know that without applying any transformation for the Model_1 we got the Higher R-squared Value i.e. 0.957 as comapare to all Model

Hence the Model_1 is better model to predict Salary_hike

THE END