**1) Delivery\_time -> Predict delivery time using sorting time**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

import statsmodels.formula.api as smf

data=pd.read\_csv("C:\\Users\\SHARAN\\Desktop\\ExcelR\\Assignment\\Assignment-4 Simple linear regression\\delivery\_time.csv")

data.rename(columns={"Delivery Time":"d","Sorting Time":"s"},inplace=True)

data.head(3)

|  | **d** | **s** |
| --- | --- | --- |
| **0** | 21.00 | 10 |
| **1** | 13.50 | 4 |
| **2** | 19.75 | 6 |

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 21 entries, 0 to 20

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 d 21 non-null float64

1 s 21 non-null int64

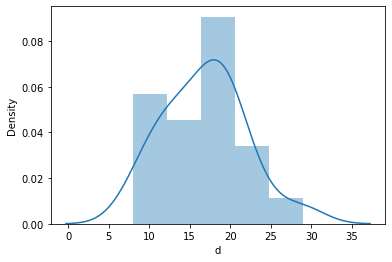
dtypes: float64(1), int64(1)

memory usage: 464.0 bytes

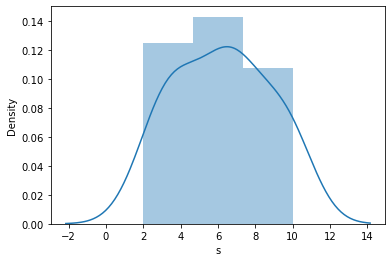
data.head(5)

|  | **d** | **s** |
| --- | --- | --- |
| **0** | 21.00 | 10 |
| **1** | 13.50 | 4 |
| **2** | 19.75 | 6 |
| **3** | 24.00 | 9 |
| **4** | 29.00 | 10 |

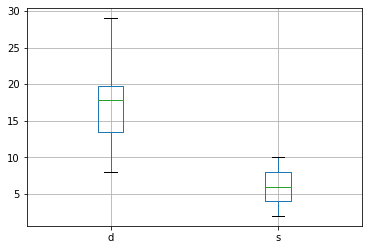
sns.distplot(data.d)



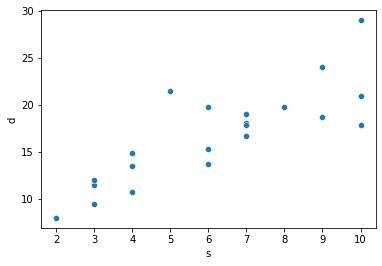
sns.distplot(data.s)



boxplot=data[["d","s"]].boxplot()



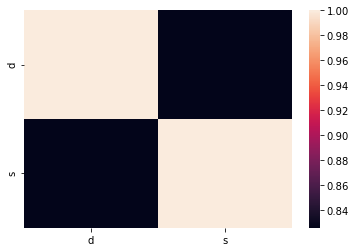
scatter=sns.scatterplot(x="s",y="d",data=data)



data.corr()

|  | **d** | **s** |
| --- | --- | --- |
| **d** | 1.000000 | 0.825997 |
| **s** | 0.825997 | 1.000000 |

sns.heatmap(data.corr())



model=smf.ols("d~np.log(s)",data=data).fit()

model.params

Intercept 1.159684

np.log(s) 9.043413

dtype: float64

model.summary()

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | d | **R-squared:** | 0.695 |
| **Model:** | OLS | **Adj. R-squared:** | 0.679 |
| **Method:** | Least Squares | **F-statistic:** | 43.39 |
| **Date:** | Sun, 12 Dec 2021 | **Prob (F-statistic):** | 2.64e-06 |
| **Time:** | 15:18:03 | **Log-Likelihood:** | -50.912 |
| **No. Observations:** | 21 | **AIC:** | 105.8 |
| **Df Residuals:** | 19 | **BIC:** | 107.9 |
| **Df Model:** | 1 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **Intercept** | 1.1597 | 2.455 | 0.472 | 0.642 | -3.978 | 6.297 |
| **np.log(s)** | 9.0434 | 1.373 | 6.587 | 0.000 | 6.170 | 11.917 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 5.552 | **Durbin-Watson:** | 1.427 |
| **Prob(Omnibus):** | 0.062 | **Jarque-Bera (JB):** | 3.481 |
| **Skew:** | 0.946 | **Prob(JB):** | 0.175 |
| **Kurtosis:** | 3.628 | **Cond. No.** | 9.08 |

model.tvalues

Intercept 0.472460

np.log(s) 6.586789

dtype: float64

model.pvalues

Intercept 0.641980

np.log(s) 0.000003

dtype: float64

data1=data.iloc[:,1]

model.predict(data1)

0 21.982913

1 13.696517

2 17.363305

3 21.030094

4 21.982913

5 17.363305

6 18.757354

7 11.094889

8 21.982913

9 21.030094

10 19.964933

11 13.696517

12 18.757354

13 11.094889

14 11.094889

15 13.696517

16 17.363305

17 18.757354

18 7.428100

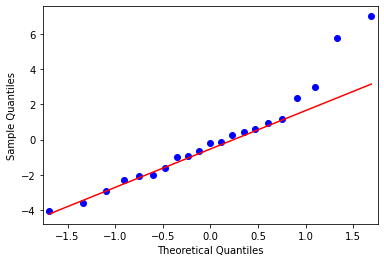
19 18.757354

20 15.714496

dtype: float64

qqplot=sm.qqplot(model.resid,line='q')

plt.show()



**2) Salary\_hike -> Build a prediction model for Salary\_hike**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.graphics.regressionplots import influence\_plot

import statsmodels.formula.api as smf

data=pd.read\_csv("C:\\Users\\SHARAN\\Desktop\\ExcelR\\Assignment\\Assignment-4 Simple linear regression\\Salary\_Data.csv")

data.head(3)

|  | **YearsExperience** | **Salary** |
| --- | --- | --- |
| **0** | 1.1 | 39343.0 |
| **1** | 1.3 | 46205.0 |
| **2** | 1.5 | 37731.0 |

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)

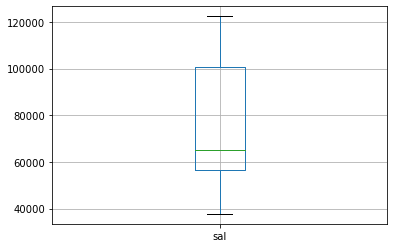
memory usage: 608.0 bytes

data.rename(columns={"YearsExperience":"exp","Salary":"sal"},inplace=True)

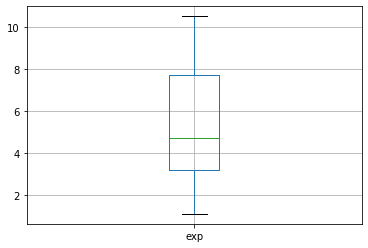
data.head(3)

|  | **exp** | **sal** |
| --- | --- | --- |
| **0** | 1.1 | 39343.0 |
| **1** | 1.3 | 46205.0 |
| **2** | 1.5 | 37731.0 |

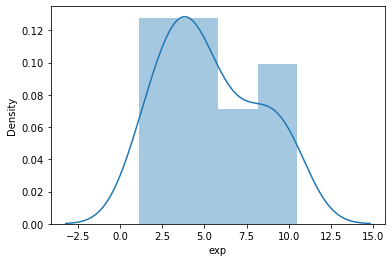
data[['sal']].boxplot()



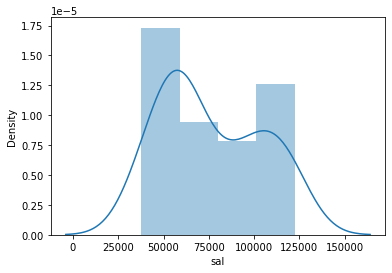
data[['exp']].boxplot()



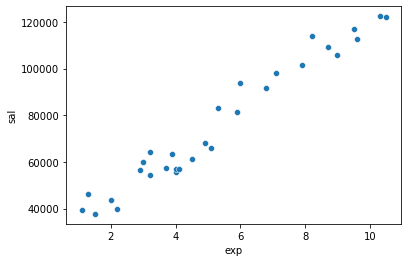
sns.distplot(data.exp)



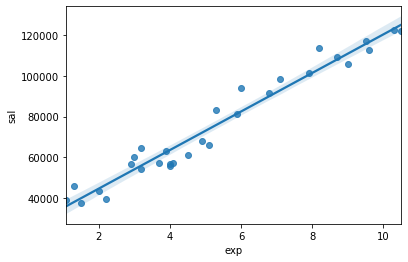
sns.distplot(data.sal)



sns.scatterplot(x='exp',y='sal',data=data)



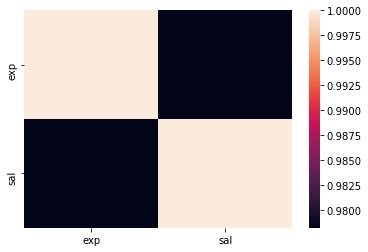
sns.regplot(x='exp',y='sal',data=data)



data.corr()

|  | **exp** | **sal** |
| --- | --- | --- |
| **exp** | 1.000000 | 0.978242 |
| **sal** | 0.978242 | 1.000000 |

sns.heatmap(data.corr())



model=smf.ols('sal~exp+np.square(exp)',data=data).fit()

model.params

Intercept 26214.932677

exp 9259.283888

np.square(exp) 16.392566

dtype: float64

model.tvalues

Intercept 0.000004

exp 0.000022

np.square(exp) 0.914983

dtype: float64

model.summary()

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | sal | **R-squared:** | 0.957 |
| **Model:** | OLS | **Adj. R-squared:** | 0.954 |
| **Method:** | Least Squares | **F-statistic:** | 300.3 |
| **Date:** | Sun, 12 Dec 2021 | **Prob (F-statistic):** | 3.59e-19 |
| **Time:** | 17:49:08 | **Log-Likelihood:** | -301.43 |
| **No. Observations:** | 30 | **AIC:** | 608.9 |
| **Df Residuals:** | 27 | **BIC:** | 613.1 |
| **Df Model:** | 2 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **Intercept** | 2.621e+04 | 4554.674 | 5.756 | 0.000 | 1.69e+04 | 3.56e+04 |
| **exp** | 9259.2839 | 1811.007 | 5.113 | 0.000 | 5543.405 | 1.3e+04 |
| **np.square(exp)** | 16.3926 | 152.121 | 0.108 | 0.915 | -295.734 | 328.520 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 2.181 | **Durbin-Watson:** | 1.649 |
| **Prob(Omnibus):** | 0.336 | **Jarque-Bera (JB):** | 1.627 |
| **Skew:** | 0.384 | **Prob(JB):** | 0.443 |
| **Kurtosis:** | 2.156 | **Cond. No.** | 223. |
| data1=data.iloc[:,:1]  model.predict(data1)  0 36419.979959  1 38279.705169  2 40140.741784  3 44799.070719  4 46664.697252  5 53204.717436  6 54140.317439  7 56012.500998  8 56012.500998  9 60698.697296  10 62575.470775  11 63514.349291  12 63514.349291  13 64453.555659  14 68213.659642  15 71979.009247  16 73863.651157  17 75749.604472  18 81415.332851  19 82360.768394  20 89936.055383  21 92782.197551  22 100386.335458  23 103243.296720  24 108011.455849  25 110876.285543  26 115657.558725  27 116614.796916  28 123324.644086  29 125244.693940  dtype: float64  qqplot=sm.qqplot(model.resid,line='q')  plt.show() |  |  |  |