In [1]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```python
data = pd.DataFrame(pd.read_csv("c:/Users/Ankit/Desktop/tsft1.csv"))
```

In [9]:
```python
data.shape
```

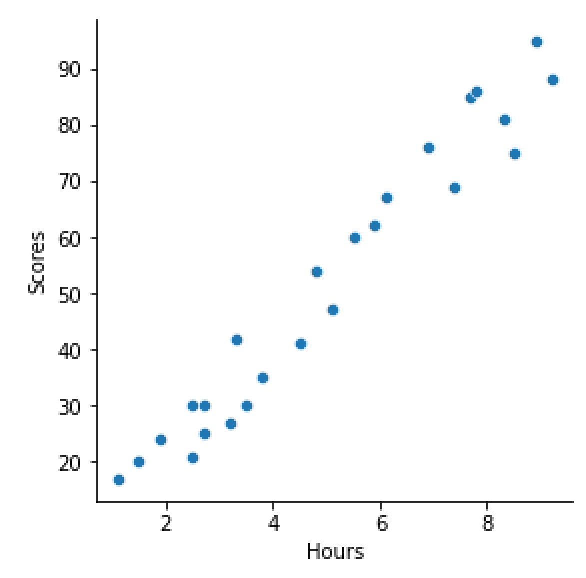
Out[9]:
```
(25, 2)
```

In [4]:
```python
data.head()
```

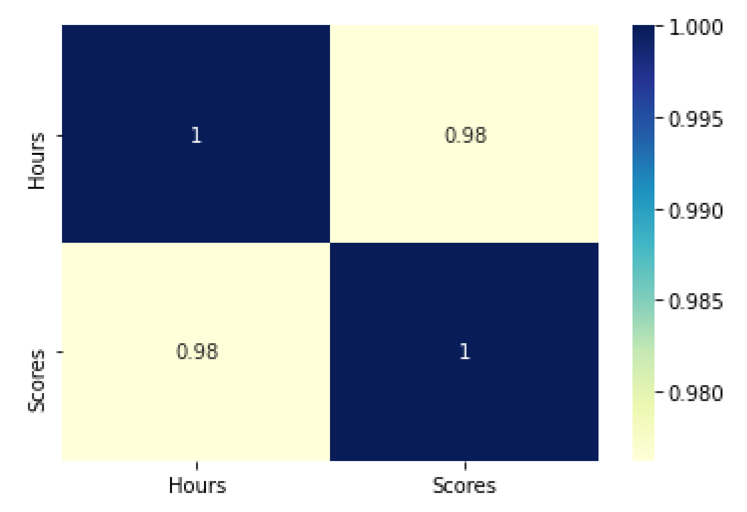
Out[4]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [5]:
```python
sns.pairplot(data, x_vars =['Hours'],y_vars = ['Scores'],height=4,aspect=1,kind='scatter')
plt.show()
```



In [6]:
```python
sns.heatmap(data.corr(), cmap="YlGnBu",annot = True)
plt.show()
```



In [12]:
```python
X = data['Hours']
Y = data['Scores']
```

In [33]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y, train_size = 0.7, test_size = 0.2, random_state = 25 )
```

In [34]:
```python
import statsmodels.api as sm
```

In [35]:
```python
x_train_sm = sm.add_constant(x_train)
```

In [36]:
```python
x_train_sm
```

Out[36]:

	const	Hours
13	1.0	3.3
0	1.0	2.5
6	1.0	9.2
16	1.0	2.5
11	1.0	5.9
20	1.0	2.7
14	1.0	1.1
23	1.0	6.9
19	1.0	7.4
3	1.0	8.5
7	1.0	5.5
1	1.0	5.1
5	1.0	1.5
24	1.0	7.8
8	1.0	8.3
18	1.0	6.1
12	1.0	4.5

In [37]:
```python
lr = sm.OLS(y_train,x_train_sm).fit()
print(lr.summary())
```

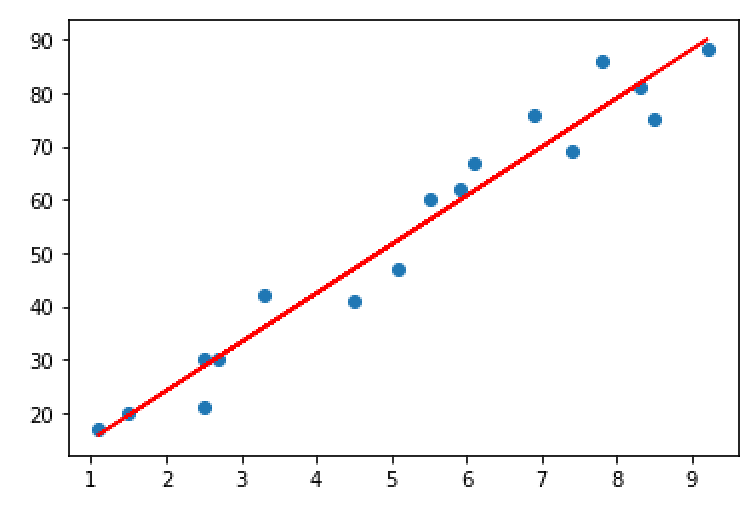
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Scores   R-squared:                       0.954
Model:                            OLS   Adj. R-squared:                  0.950
Method:                 Least Squares   F-statistic:                     307.9
Date:                Mon, 08 Nov 2021   Prob (F-statistic):           2.08e-11
Time:                        13:58:59   Log-Likelihood:                -51.761
No. Observations:                  17   AIC:                             107.5
Df Residuals:                      15   BIC:                             109.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.8821      3.022      1.946      0.071      -0.559      12.323
Hours          9.1442      0.521     17.547      0.000       8.033      10.255
==============================================================================
Omnibus:                        1.034   Durbin-Watson:                   2.353
Prob(Omnibus):                  0.596   Jarque-Bera (JB):                0.744
Skew:                          -0.060   Prob(JB):                        0.689
Kurtosis:                       1.982   Cond. No.                         13.7
==============================================================================

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

In [38]:
```python
lr.params
```

Out[38]:
```
const    5.882081
Hours    9.144196
dtype: float64
```

In [39]:
```python
#plotting best fit line gibven by model
#plotting this line y = 6.88 + .05 x
plt.scatter(x_train,y_train)
plt.plot(x_train,5.88 + 9.14*x_train, 'r')
plt.show
```

Out[39]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

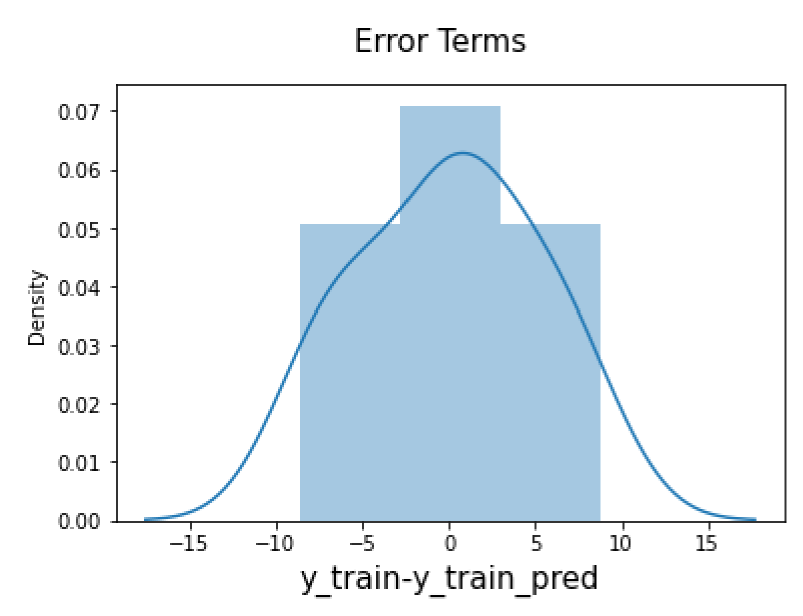


In [40]:
```python
#model evaluation
#distribution of error terms
#we need to check that error termws are normally distributed or not
#we can check it by plotting histogram of errors
```

In [41]:
```python
y_train_pred = lr.predict(x_train_sm)
error =(y_train-y_train_pred)
```

In [42]:
```python
fig = plt.figure()
sns.distplot(error)
fig.suptitle('Error Terms',fontsize=15)
plt.xlabel('y_train-y_train_pred',fontsize=15)
plt.show
```

Out[42]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [43]:
```python
#here we can see that errors are following normal distribution with mean = 0 # all ok
```

In [44]:
```python
#prediction on test set
```

In [45]:
```python
x_test_sm = sm.add_constant(x_test)
y_test_pred = lr.predict(x_test_sm)
```

In [46]:
```python
y_test_pred.head()
```

Out[46]:
```
2     35.143509
9     30.571411
17    23.256054
10    76.292392
21    49.774223
dtype: float64
```

In [47]:
```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

In [48]:
```python
#RMSE
np.sqrt(mean_squared_error(y_test,y_test_pred))
```

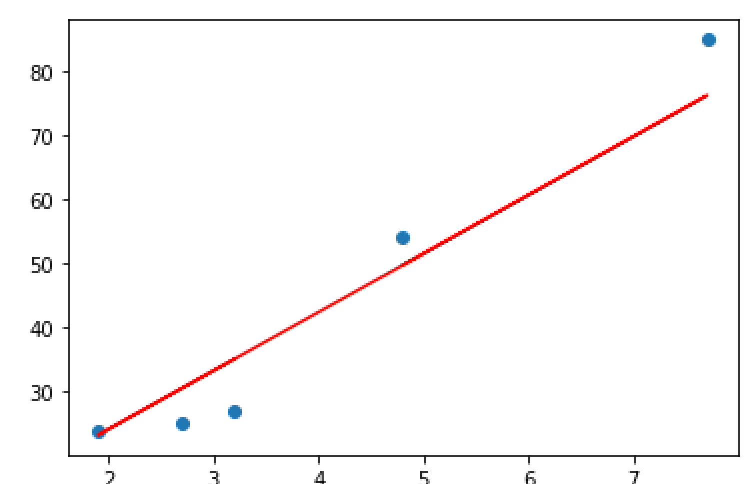
Out[48]:
```
6.190160581267405
```

In [49]:
```python
#rsqaured
r_squared = r2_score(y_test,y_test_pred)
r_squared
```

Out[49]:
```
0.9322043736343297
```

In [51]:
```python
#visualising the fit on test set
plt.scatter(x_test,y_test)
plt.plot(x_test,5.88 + 9.14*x_test, 'r')
plt.show
```

Out[51]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [52]:
```python
# as r2 for train set is .94 and r2 for test st is .93 Therefore this is a good model
```

In [ ]: