GRIP: The spark foundation

Data science and Business analytics intern

Author : jagrati

task 1: prediction using Supervised ML

In this task we have to predict the percentage score of a student based on the number of hour studied. The task has two variables where the feature is the no. of hours studied and the target value is the percentage score . This can be solved using sinple linear regression.

In [2]:
```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading data from remote url

In [3]:
```
url="http://bit.ly/w-data"
data=pd.read_csv(url)
```

Exploring data

In [4]:
```
print (data.shape)
data.head()
```

```
(25, 2)
```

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]:
```
data.describe()
```

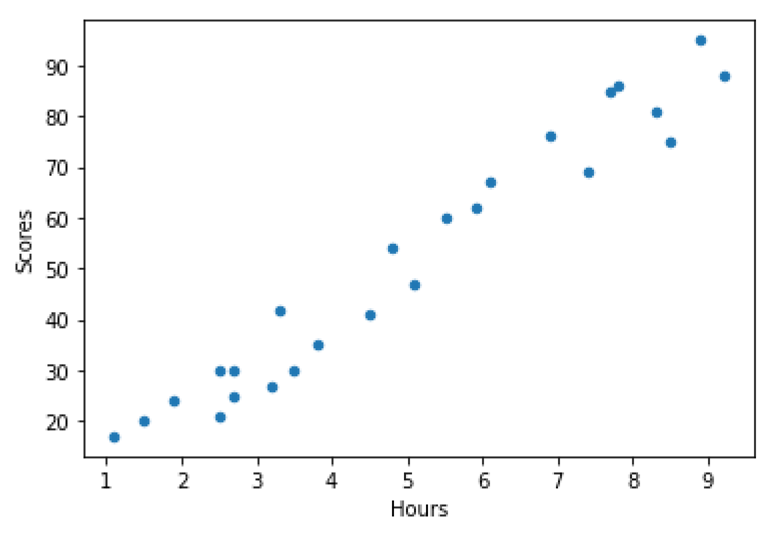
Out[5]:

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [9]:
```
data.plot(kind='scatter',x='Hours',y='Scores');
plt.show()
```



In [10]:
```
data.corr(method="pearson")
```

Out[10]:

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [11]:
```
data.corr(method="spearman")
```

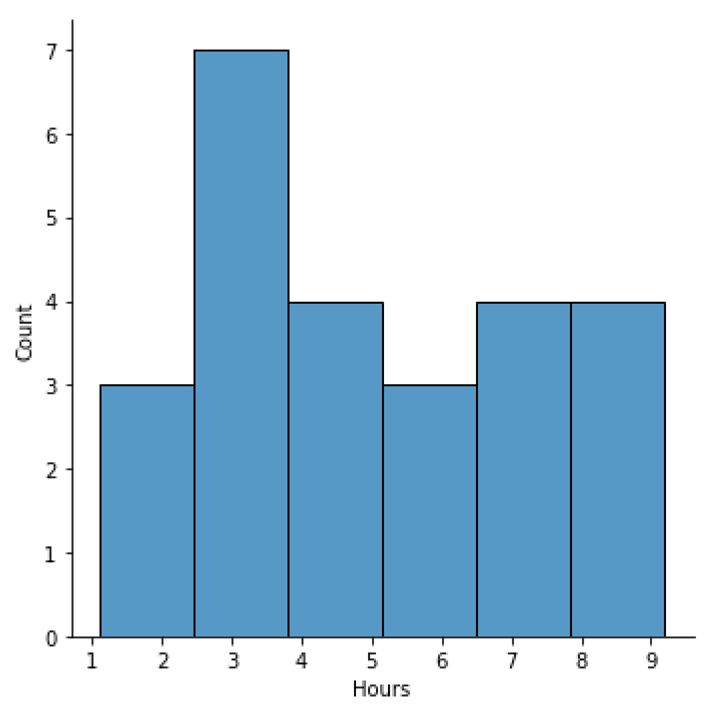
Out[11]:

	Hours	Scores
Hours	1.000000	0.971891
Scores	0.971891	1.000000

In [12]:
```
hours=data['Hours']
scores=data['Scores']
```

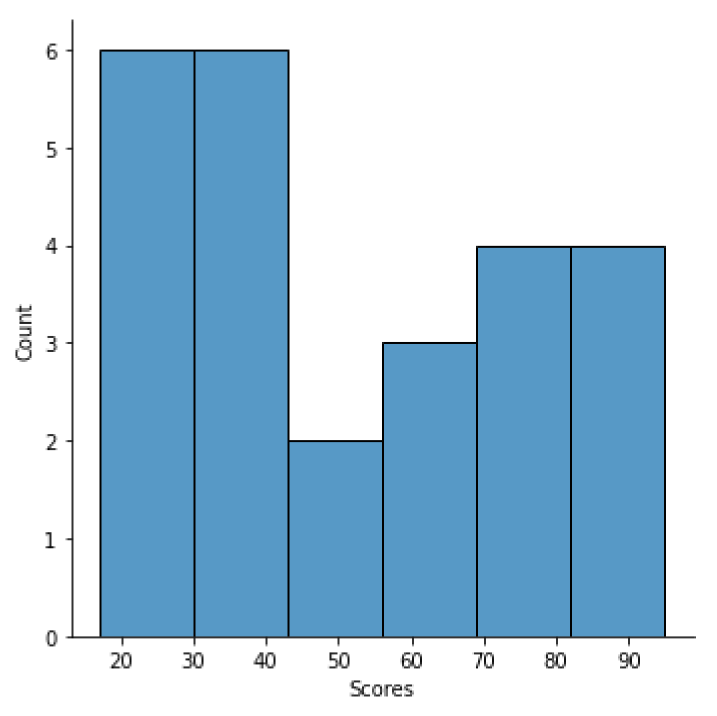
In [14]:
```
sns.displot(hours)
```

Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x205dd9da7c0>
```



In [15]:
```
sns.displot(scores)
```

Out[15]:
```
<seaborn.axisgrid.FacetGrid at 0x205dda53af0>
```



Linear Regression

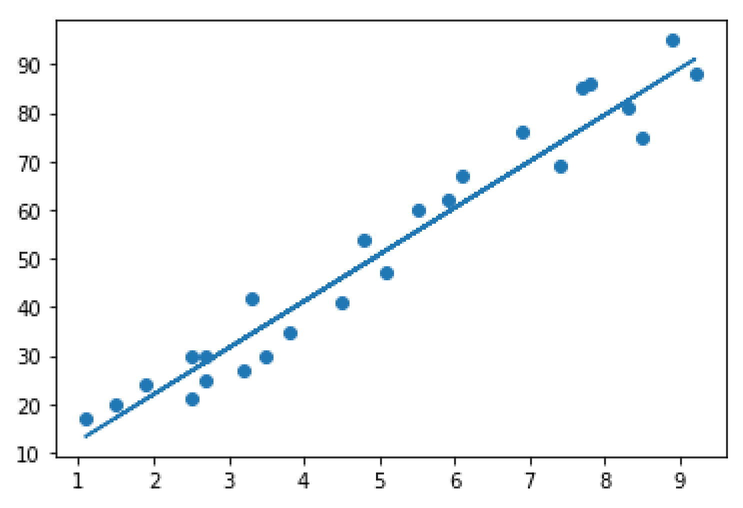
In [18]:
```
X=data.iloc[:,:-1].values
Y=data.iloc[:, 1].values
```

In [25]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(X,Y,test_size=0.2,random_state=50)
```

In [26]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train,Y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```
m=reg.coef_
c=reg.intercept_
line=m*X+c
plt.scatter(X,Y)
plt.plot(X,line);
plt.show()
```



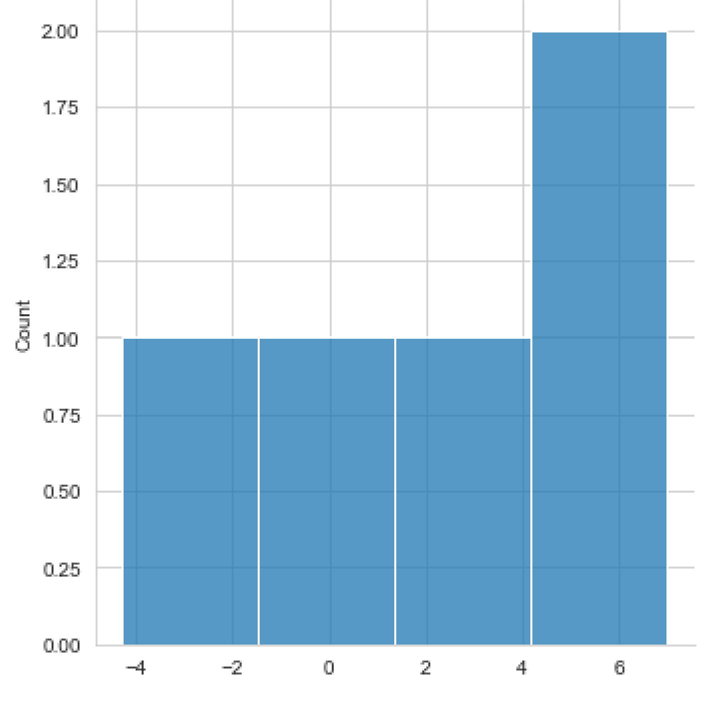
In [29]:
```
Y_pred=reg.predict(X_test)
```

In [30]:
```
actual_predicted=pd.DataFrame({'Target':Y_test,'predicted':Y_pred})
actual_predicted
```

Out[30]:

	Target	predicted
0	95	88.211394
1	30	28.718453
2	76	69.020122
3	35	39.273652
4	17	13.365436

In [31]:
```
sns.set_style('whitegrid')
sns.displot(np.array(Y_test-Y_pred))
plt.show()
```



what would be the predicted score if a student studies for 9.25 hours/day?

In [32]:
```
h=9.25
s=reg.predict([[h]])
print("if a student studies for {} hours per day he/she will score {} % in exam.".format(h,s))
```

```
if a student studies for 9.25 hours per day he/she will score [91.56986604] % in exam.
```

Model Evaluation

In [34]:
```
from sklearn import metrics
from sklearn.metrics import r2_score
print('mean absolute error:',metrics.mean_absolute_error(Y_test,Y_pred))
print('R2 score:',r2_score(Y_test,Y_pred))
```

```
mean absolute error: 4.5916495300630285
R2 score: 0.971014141329942
```

In [ ]: