LINEAR REGRESSION MODEL ON STUDENTS PERFORMANCE

The dataset was obtained from Kaggle platform.The purpose of this analysis was to predict Student's marks based on time of study and number of courses.
Linear Regression had r2_squared score of 94% which is good prediction.
This implies Marks are greatly affected by time of study and number of courses taken by a student.

In [1]:
```
#importing relevant libraries
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [2]:
```
#loading dataset
data=pd.read_csv('Desktop/students_exam_marks.csv')
```

In [3]:
```
#head of data
data.head()
```

Out[3]:

	number_courses	time_study	Marks
0	3	4.508	19.202
1	4	0.096	7.734
2	4	3.133	13.811
3	6	7.909	53.018
4	8	7.811	55.299

In [4]:
```
#descriptive statistics
data.describe()
```

Out[4]:

	number_courses	time_study	Marks
count	100.000000	100.000000	100.000000
mean	5.290000	4.077140	24.417690
std	1.799523	2.372914	14.326199
min	3.000000	0.096000	5.609000
25%	4.000000	2.058500	12.633000
50%	5.000000	4.022000	20.059500
75%	7.000000	6.179250	36.676250
max	8.000000	7.957000	55.299000

In [5]:
```
#data info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   number_courses  100 non-null    int64
 1   time_study      100 non-null    float64
 2   Marks           100 non-null    float64
dtypes: float64(2), int64(1)
memory usage: 2.5 KB
```

In [6]:
```
#shape of data
data.shape
```

Out[6]:
```
(100, 3)
```

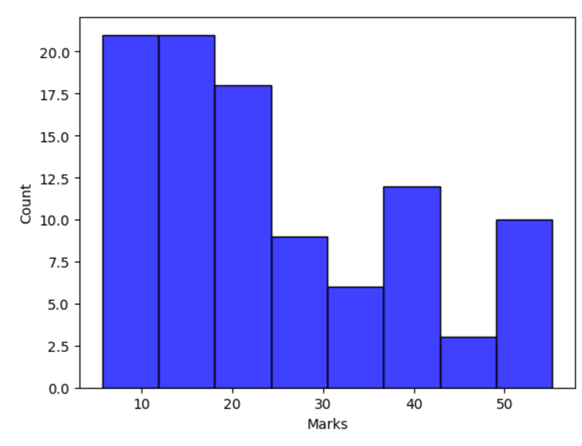
In [7]:
```
#checking null values
data.isnull().sum()
```

Out[7]:
```
number_courses    0
time_study        0
Marks             0
dtype: int64
```

EXPLONATORY DATA ANALYSIS

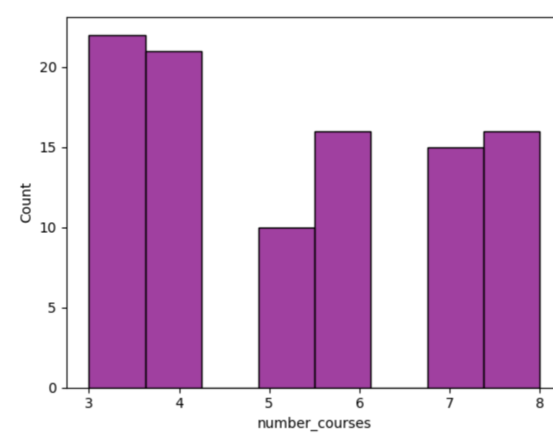
In [8]:
```
#histplot on marks
sns.histplot(data['Marks'],color='blue')
```

Out[8]:
```
<Axes: xlabel='Marks', ylabel='Count'>
```



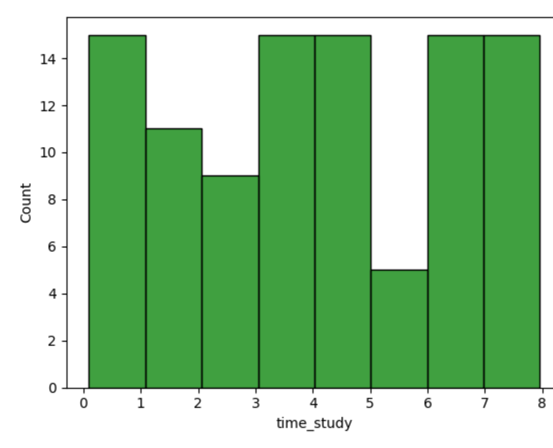
In [9]:
```
#histplot on number of courses
sns.histplot(data['number_courses'],color='purple')
```

Out[9]:
```
<Axes: xlabel='number_courses', ylabel='Count'>
```



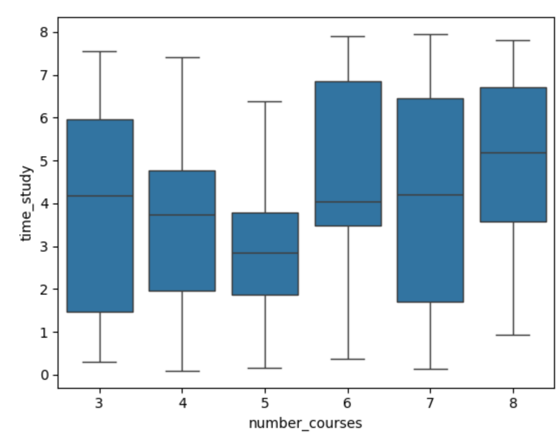
In [10]:
```
#histplot on time of study
sns.histplot(data['time_study'],color='green')
```

Out[10]:
```
<Axes: xlabel='time_study', ylabel='Count'>
```



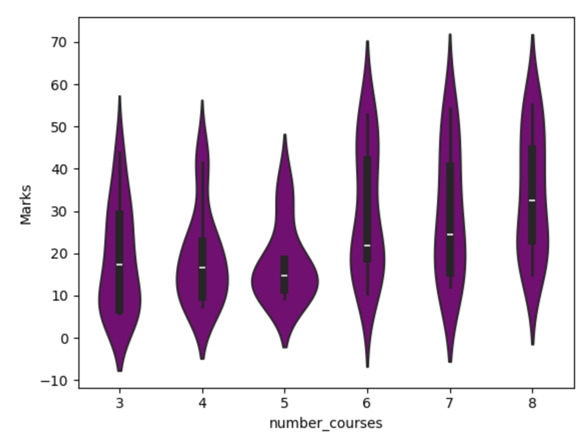
In [11]:
```
#boxplot on number_course against time of study
sns.boxplot(x='number_courses',y='time_study',data=data)
```

Out[11]:
```
<Axes: xlabel='number_courses', ylabel='time_study'>
```



In [12]:
```
#violinplot on number_course against time of study
sns.violinplot(x='number_courses',y='Marks',data=data,color='purple')
```

Out[12]:
```
<Axes: xlabel='number_courses', ylabel='Marks'>
```



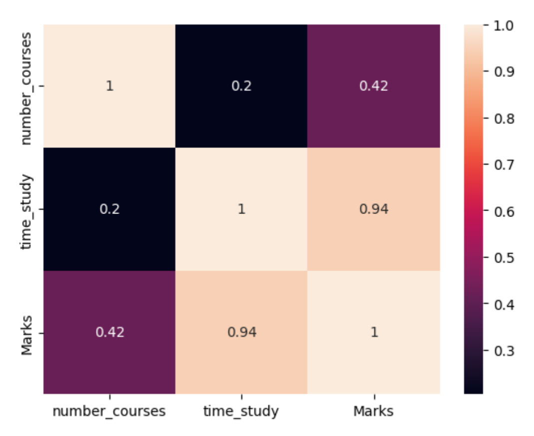
In [13]:
```
#correlation
data.corr()
```

Out[13]:

	number_courses	time_study	Marks
number_courses	1.000000	0.204844	0.417335
time_study	0.204844	1.000000	0.942254
Marks	0.417335	0.942254	1.000000

In [14]:
```
sns.heatmap(data.corr(),annot=True)
```

Out[14]:
```
<Axes: >
```



In [15]:
```
## MODEL TRAINING
```

In [16]:
```
#declare variables
X=data[['number_courses','time_study']]
y=data['Marks']
```

In [17]:
```
from sklearn.preprocessing import StandardScaler
```

In [18]:
```
#  standardization
scaler=StandardScaler()   # declaring standard scaler
scaler.fit(X)
```

Out[18]:
```
▾ StandardScaler
StandardScaler()
```

In [19]:
```
#scaling each feature
X_scaled=scaler.transform(X)
```

LINEAR REGRESSION MODEL

In [20]:
```
##STATSMODELS
```

In [21]:
```
import statsmodels.api as sm
```

In [22]:
```
x=sm.add_constant(X)
results=sm.OLS(y,x).fit()
results.summary()
```

Out[22]:

OLS Regression Results

Dep. Variable:	Marks	**R-squared:**	0.940
Model:	OLS	**Adj. R-squared:**	0.939
Method:	Least Squares	**F-statistic:**	764.8
Date:	Thu, 14 Dec 2023	**Prob (F-statistic):**	4.09e-60
Time:	15:35:04	**Log-Likelihood:**	-266.62
No. Observations:	100	**AIC:**	539.2
Df Residuals:	97	**BIC:**	547.1
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-7.4563	1.174	-6.349	0.000	-9.787	-5.125
number_courses	1.8641	0.202	9.243	0.000	1.464	2.264
time_study	5.3992	0.153	35.303	0.000	5.096	5.703

Omnibus:	29.529	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9.956
Skew:	0.526	**Prob(JB):**	0.00689
Kurtosis:	1.867	**Cond. No.**	23.9

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

In [23]:
```
##SKLEARN
```

In [24]:
```
#regression with scaled features
reg=LinearRegression()
reg.fit(X,y)
```

Out[24]:
```
▾ LinearRegression
LinearRegression()
```

In [25]:
```
reg.score(X,y)
```

Out[25]:
```
0.9403656320238896
```

In [26]:
```
#R squared value
r_squared=reg.score(X,y)
r_squared
```

Out[26]:
```
0.9403656320238896
```

In [27]:
```
#declaring number of observations and number of independent variables
n=X.shape[0]
k=x.shape[1]
```

In [28]:
```
#adjusted_r_squared
adjusted_r_squared=1-(1-r_squared)*((n-1)/(n-k-1))
adjusted_r_squared
```

Out[28]:
```
0.9385020580246362
```

In [29]:
```
# constant/y_intercept/biases
reg.intercept_
```

Out[29]:
```
-7.456346231178355
```

In [30]:
```
# coefficiencies(weights)
reg.coef_
```

Out[30]:
```
array([1.86405074, 5.39917879])
```

In [31]:
```
#importing f_regression
from sklearn.feature_selection import f_regression
```

In [32]:
```
# f_statistic values
F_statistics=f_regression(X,y)[0]
F_statistics
```

Out[32]:
```
array([ 20.66822463, 775.77043264])
```

In [33]:
```
#p_values
P_values=f_regression(X,y)[1].round(4)
P_values
```

Out[33]:
```
array([0., 0.])
```

In [34]:
```
reg_summary=pd.DataFrame(['Intercept','number_courses','time_study'],columns=['Features'])
reg_summary['weights']=reg.intercept_,reg.coef_[0],reg.coef_[1]
```

In [35]:
```
reg_summary
```

Out[35]:

	Features	weights
0	Intercept	-7.456346
1	number_courses	1.864051
2	time_study	5.399179

In [36]:
```
#creating columns name of summary table
reg_summary=pd.DataFrame(data=X.columns.values,columns=['features'])
```

In [37]:
```
#Table of summary statistics
reg_summary['coefficies']=reg.coef_
reg_summary['intercept']=reg.intercept_
reg_summary['p_values']=f_regression(X,y)[1].round(4)
reg_summary['F_statistics']=f_regression(X,y)[0]
reg_summary['r_squared']=reg.score(X,y)
reg_summary['adjusted_r_squared']=1-(1-r_squared)*((n-1)/(n-k-1))
```

In [38]:
```
reg_summary
```

Out[38]:

	features	coefficies	intercept	p_values	F_statistics	r_squared	adjusted_r_squared
0	number_courses	1.864051	-7.456346	0.0	20.668225	0.940366	0.938502
1	time_study	5.399179	-7.456346	0.0	775.770433	0.940366	0.938502

In [39]:
```
#predicted data using regression equation
data_predicted=reg.predict(X)
```

In [40]:
```
#Predicted_table
Predicted_table=pd.DataFrame(columns=['predicted','Marks'])
```

In [41]:
```
#Table of original y_values against predicted
Predicted_table['predicted']=data_predicted
Predicted_table['Marks']=data['Marks']
```

In [42]:
```
Predicted_table.head(10)
```

Out[42]:

	predicted	Marks
0	22.475304	19.202
1	0.518178	7.734
2	16.915484	13.811
3	46.430063	53.018
4	49.629045	55.299
5	21.064721	17.822
6	30.871027	29.889
7	20.291305	17.264
8	23.810235	20.348
9	31.464937	30.862

In [43]:
```
#predicting values using standardized coefficients
Predicted=reg.predict(X)
```

In [44]:
```
dataset_with_predictions = pd.concat([data, pd.Series(Predicted,name='Predicted_Values')], axis=1)
```

dataset_with_predictions

In [45]:
```
#Model evaluation
```

In [46]:
```
from sklearn.metrics import mean_squared_error
```

In [47]:
```
mse=mean_squared_error(y,Predicted)
mse
```

Out[47]:
```
12.116962069108952
```

CONCLUSION

Courses and time of study explained 94% of variability in Marks obtained by students in this analysis.Hence the linear regression the model was good in prediction.

In [ ]: