OTT REVENUE PREDICTION MODEL

A model to predict the revenue in million dollars based on the number of subscribers.

Data set:

Independant variable X: Subscribers/Year/Content Spend/Profit

Dependant variable Y: Overall revenue generated in dollars

In [1]:

```
#import required libraries
import numpy as np
import pandas as pd
import os
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
#List all files used
for dirname, _, filenames in os.walk(r"C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel"):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\ContentSpend.csv
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\NumSubscribers.csv
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\NumSubscribersByRegion.csv
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\Profit.csv
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\Revenue.csv
C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\RevenueByRegion.csv
```

In [3]:

```
#import datset
df_profit = pd.read_csv(r"C:\Users\vdp10002\OneDrive - Advanced Micro Devices Inc\IISC_project\MainProject_OTTRe
df_subscribers = pd.read_csv(r"C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\NumSubscr
df_revenue = pd.read_csv(r"C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\Revenue.csv")
df_ContentSpend=pd.read_csv(r"C:\Users\vdp10002\Desktop\MainProject_OTTRevenuePredictionModel\Netflix\ContentSpe
#initial exploration of data
df_profit
```

Out[3]:

	Year	Profit
0	2012	0.050
1	2013	0.228
2	2014	0.403
3	2015	0.306
4	2016	0.379
5	2017	0.839
6	2018	1.600
7	2019	2.600
8	2020	4.500

In [4]:
```
df_subscribers
```

Out[4]:

	Year	Subscribers
0	2011	21.5
1	2012	25.7
2	2013	35.6
3	2014	47.9
4	2015	62.7
5	2016	79.9
6	2017	99.0
7	2018	124.3
8	2019	151.5
9	2020	192.9

In [5]:
```
df_revenue
```

Out[5]:

	Year	Revenue
0	2011	3.1
1	2012	3.5
2	2013	4.3
3	2014	5.4
4	2015	6.7
5	2016	8.8
6	2017	11.6
7	2018	15.7
8	2019	20.1
9	2020	24.9

In [6]:
```
df_ContentSpend
```

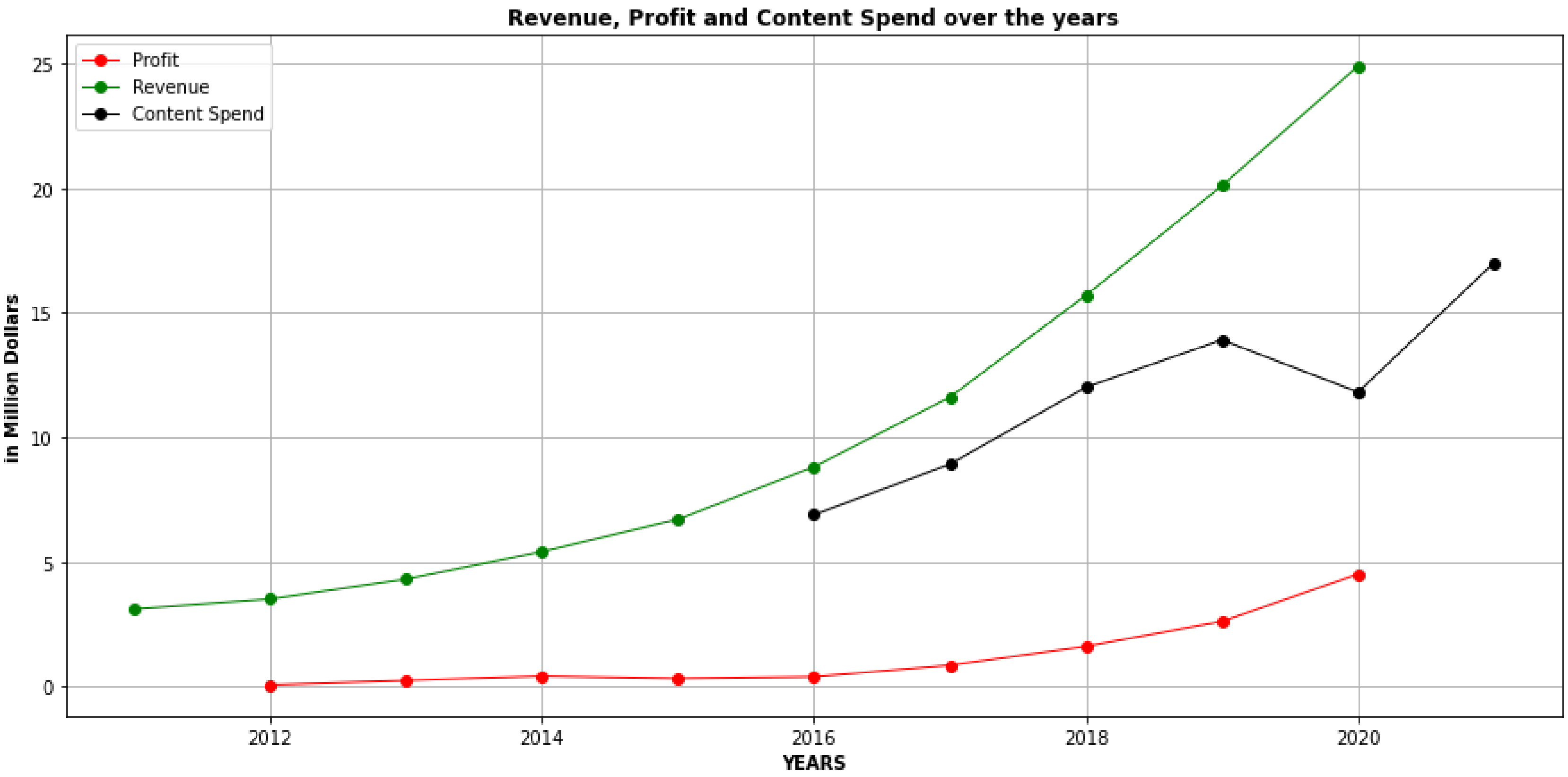
Out[6]:

	Year	Content_spend
0	2016	6.88
1	2017	8.91
2	2018	12.00
3	2019	13.90
4	2020	11.80
5	2021	17.00

VISUALIZE DATSET

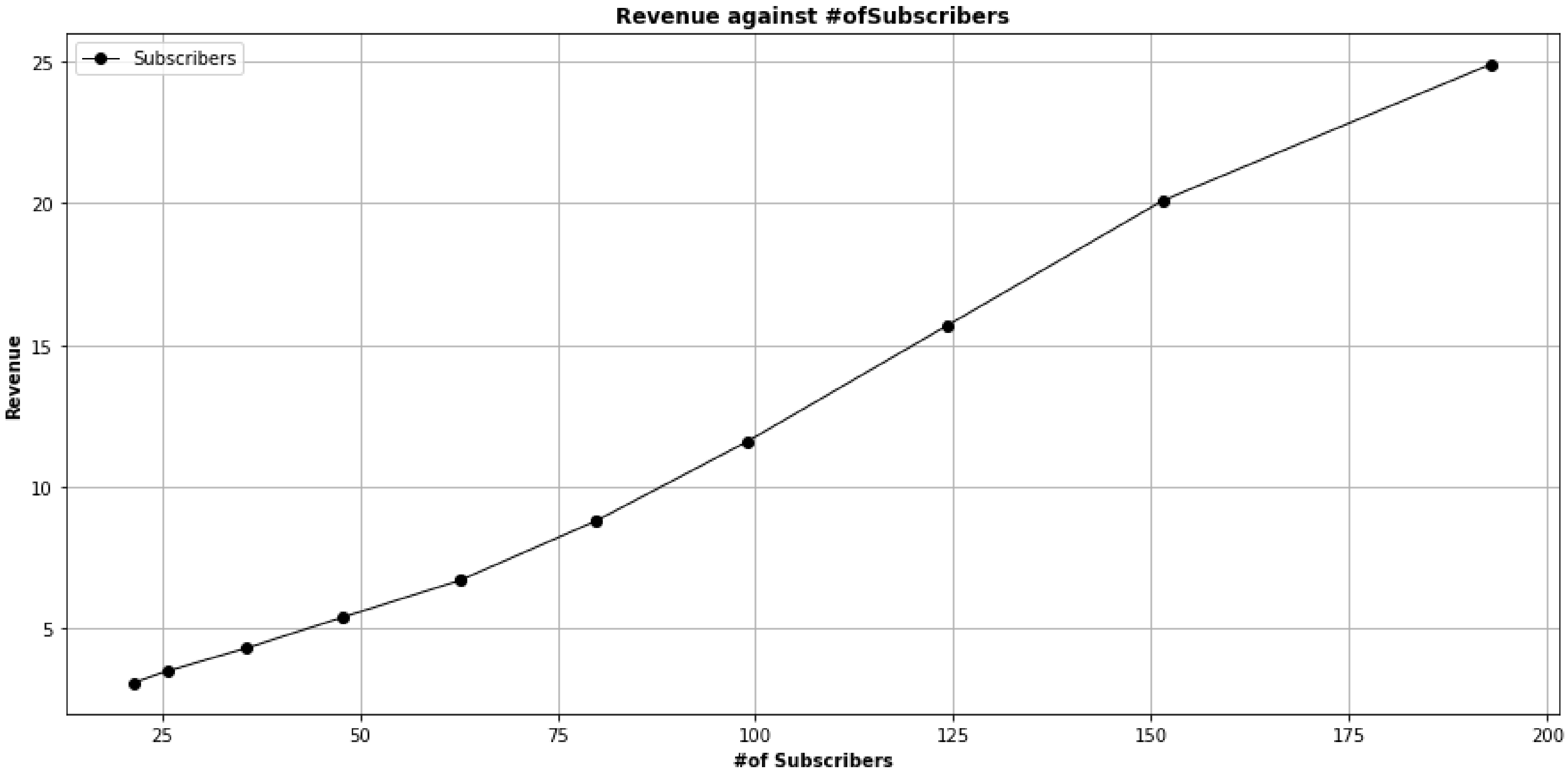
In [7]:

```
#understanding the trend of Revenue over the years, profit and content spend
x1 = df_profit['Year'].values
y1 = df_profit['Profit'].values

x2=df_subscribers['Year'].values
y2=df_subscribers['Subscribers'].values

x3=df_revenue['Year'].values
y3=df_revenue['Revenue'].values

x4=df_ContentSpend['Year'].values
y4=df_ContentSpend['Content_spend'].values

plt.rcParams["figure.figsize"] = (15,7)

plt.plot(x1, y1, 'red', label='Profit',  marker='o', linestyle='-', linewidth='1')
#plt.plot(x2, y2, 'blue', label='Subscribers')
plt.plot(x3, y3, 'green', label='Revenue',  marker='o', linestyle='-', linewidth='1')
plt.plot(x4, y4, 'black', label='Content Spend',  marker='o', linestyle='-', linewidth='1')

plt.grid()
plt.xlabel('YEARS', fontweight="bold")
plt.ylabel('in Million Dollars', fontweight="bold")
plt.title('Revenue, Profit and Content Spend over the years', fontweight="bold")
plt.legend()
plt.show()
```


Revenue, Profit and Content Spend over the years
Profit
Revenue
Content Spend
in Million Dollars
0
5
10
15
20
25
2012
2014
2016
2018
2020
YEARS

In [8]:

```
#understanding the trend of Revenue wrt # of Subscribers
x5=df_subscribers['Subscribers'].values
y5=df_revenue['Revenue'].values


plt.rcParams["figure.figsize"] = (15,7)

plt.plot(x5, y5, 'black', label='Subscribers', marker='o', linestyle='-', linewidth='1')

plt.grid()
plt.xlabel('#of Subscribers', fontweight="bold")
plt.ylabel('Revenue', fontweight="bold")
plt.title('Revenue against #ofSubscribers', fontweight="bold")
plt.legend()
plt.show()
```

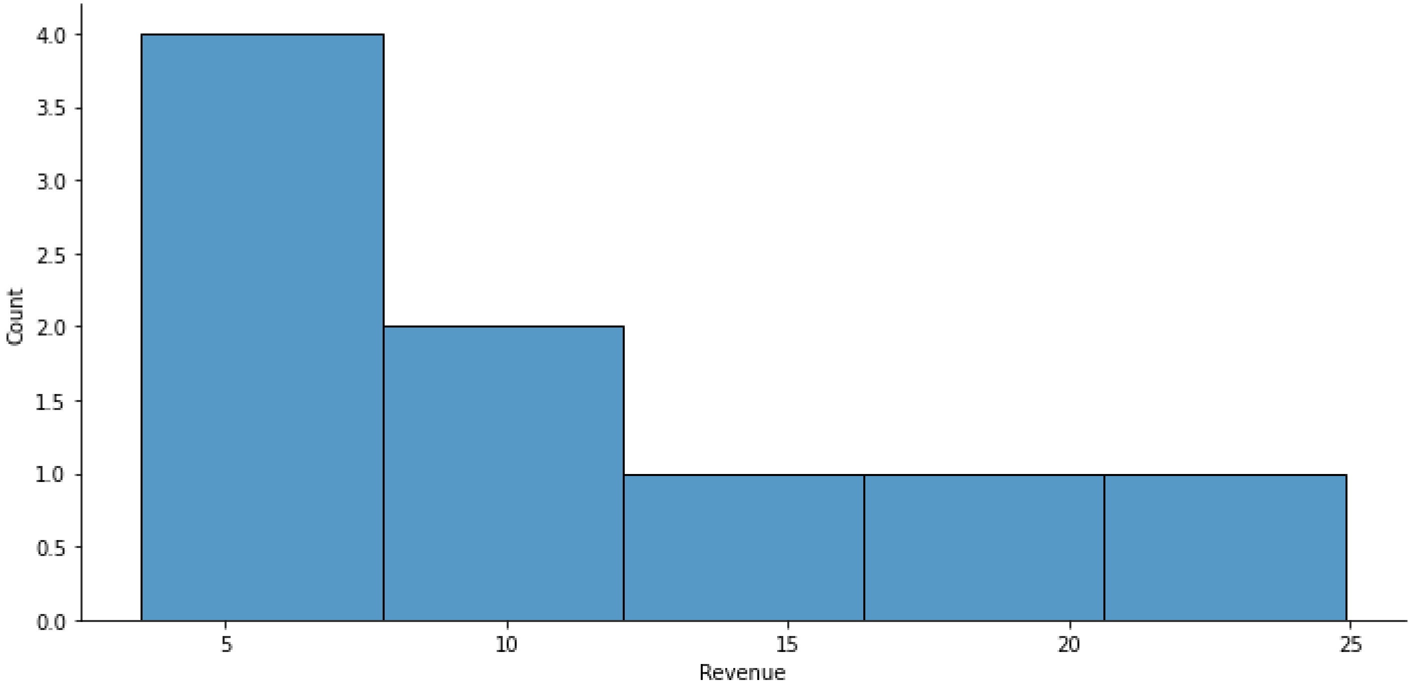


In [9]:

```
df_new = pd.merge(pd.merge(df_profit,df_revenue,on='Year'),df_subscribers,on='Year', how='right')
df_new1 = pd.merge(df_new, df_ContentSpend, on='Year', how='outer')
```

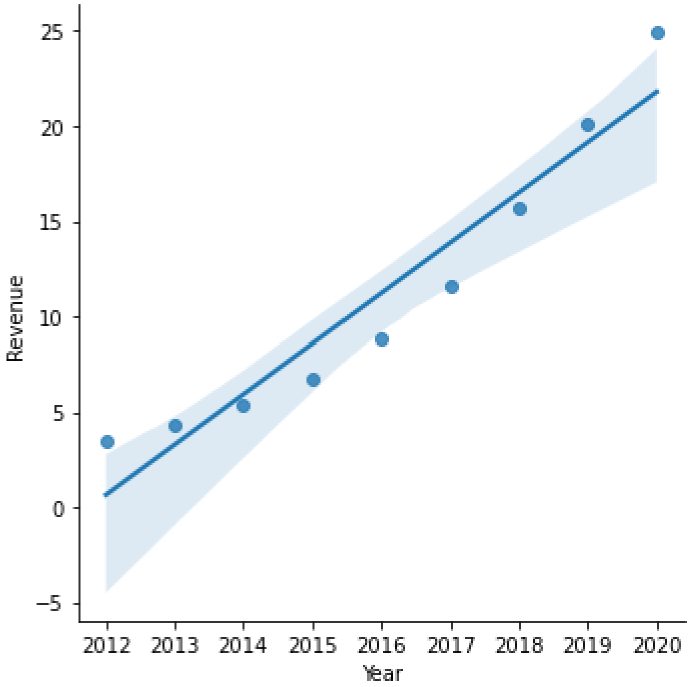
In [10]:

```
# developing a histogram using DISPLOT
sns.displot(data   = df_new1,
            x      = 'Revenue',
            height = 5,
            aspect = 2)


plt.show()
```



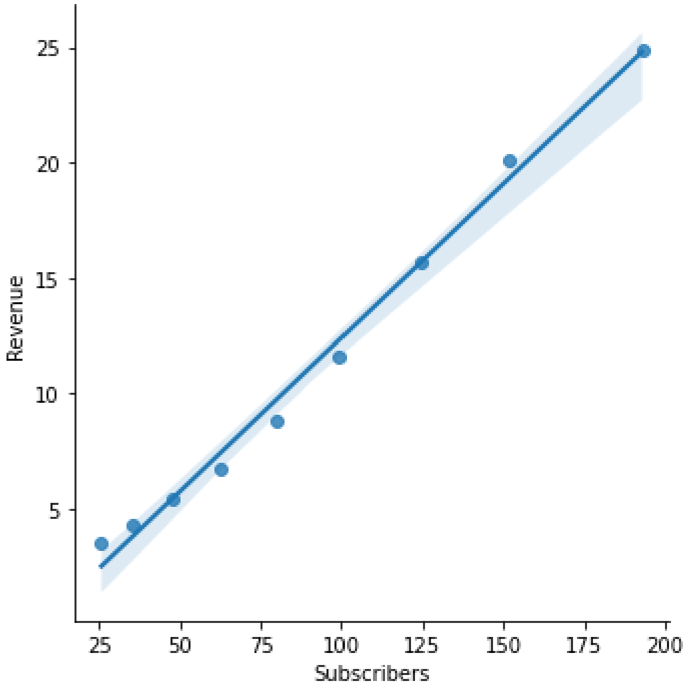
```
In [11]: sns.lmplot(x='Year', y='Revenue', data=df_new1)
```

```
Out[11]: <seaborn.axisgrid.FacetGrid at 0x247c7b34ac0>
```



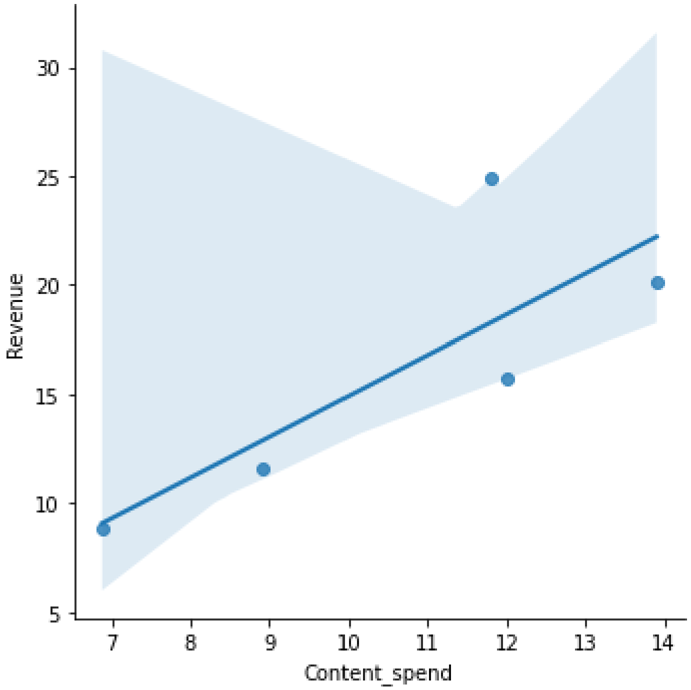
```
In [12]: sns.lmplot(x='Subscribers', y='Revenue', data=df_new1)
```

```
Out[12]: <seaborn.axisgrid.FacetGrid at 0x247c2437a00>
```



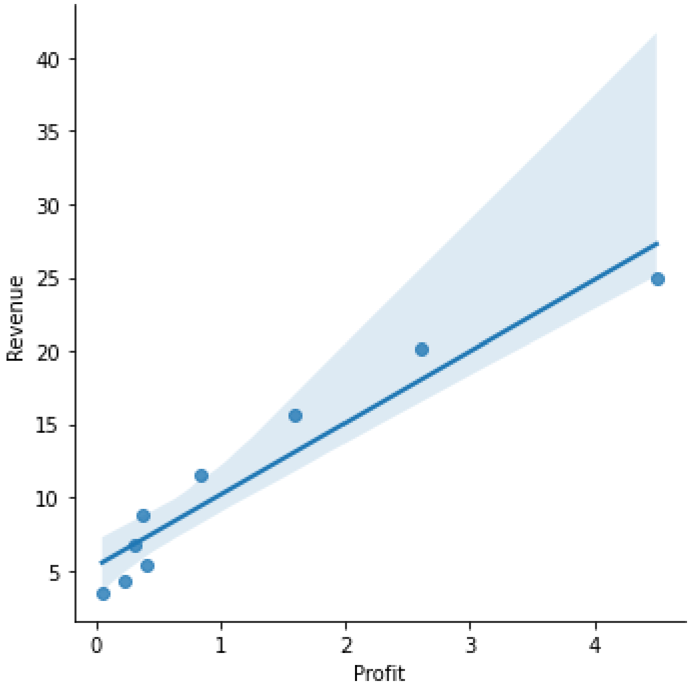
```
In [13]: sns.lmplot(x='Content_spend', y='Revenue', data=df_new1)
```

```
Out[13]: <seaborn.axisgrid.FacetGrid at 0x247c842acd0>
```



```
In [14]: sns.lmplot(x='Profit', y='Revenue', data=df_new1)
```

```
Out[14]: <seaborn.axisgrid.FacetGrid at 0x247c842aa90>
```



MISSING VALUE ANALYSIS AND IMPUTATION

In [15]:
```
df_new1.isnull()
```

Out[15]:

	Year	Profit	Revenue	Subscribers	Content_spend
0	False	True	True	False	True
1	False	False	False	False	True
2	False	False	False	False	True
3	False	False	False	False	True
4	False	False	False	False	True
5	False	False	False	False	False
6	False	False	False	False	False
7	False	False	False	False	False
8	False	False	False	False	False
9	False	False	False	False	False
10	False	True	True	True	False

In [16]:
```
df_new1
```

Out[16]:

	Year	Profit	Revenue	Subscribers	Content_spend
0	2011	NaN	NaN	21.5	NaN
1	2012	0.050	3.5	25.7	NaN
2	2013	0.228	4.3	35.6	NaN
3	2014	0.403	5.4	47.9	NaN
4	2015	0.306	6.7	62.7	NaN
5	2016	0.379	8.8	79.9	6.88
6	2017	0.839	11.6	99.0	8.91
7	2018	1.600	15.7	124.3	12.00
8	2019	2.600	20.1	151.5	13.90
9	2020	4.500	24.9	192.9	11.80
10	2021	NaN	NaN	NaN	17.00

In [17]:

```
## Replace all NaN values with 0
#df_new2= df_new1.fillna(0)
#df_new2
df_new2=df_new1
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.NaN, strategy='median')
print(imputer)
df_new2.Profit = imputer.fit_transform(df_new2['Profit'].values.reshape(-1,1))
df_new2.Revenue = imputer.fit_transform(df_new2['Revenue'].values.reshape(-1,1))
df_new2.Subscribers = imputer.fit_transform(df_new2['Subscribers'].values.reshape(-1,1))
df_new2.Content_spend = imputer.fit_transform(df_new2['Content_spend'].values.reshape(-1,1))
df_new2
```

```
SimpleImputer(strategy='median')
```

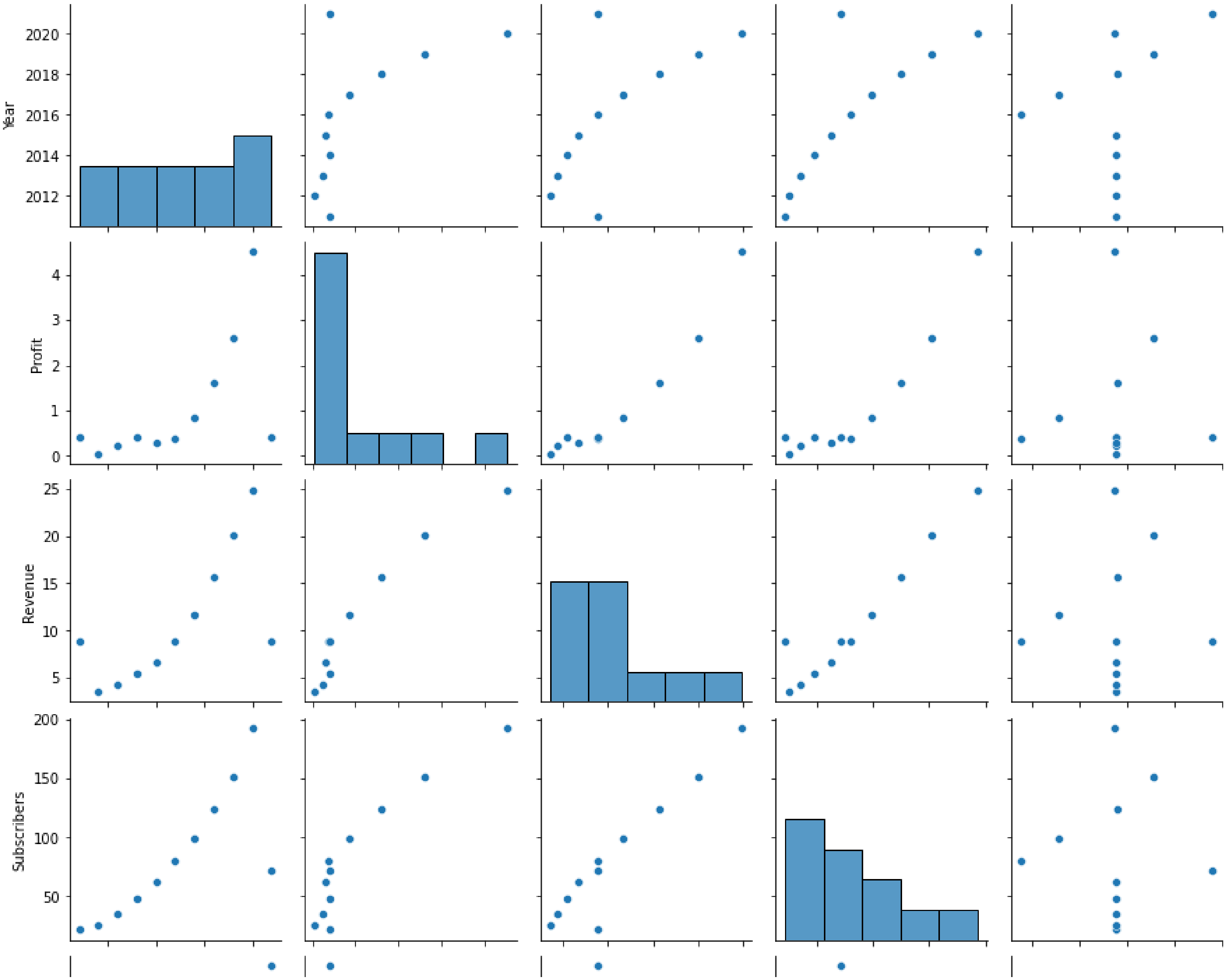
Out[17]:

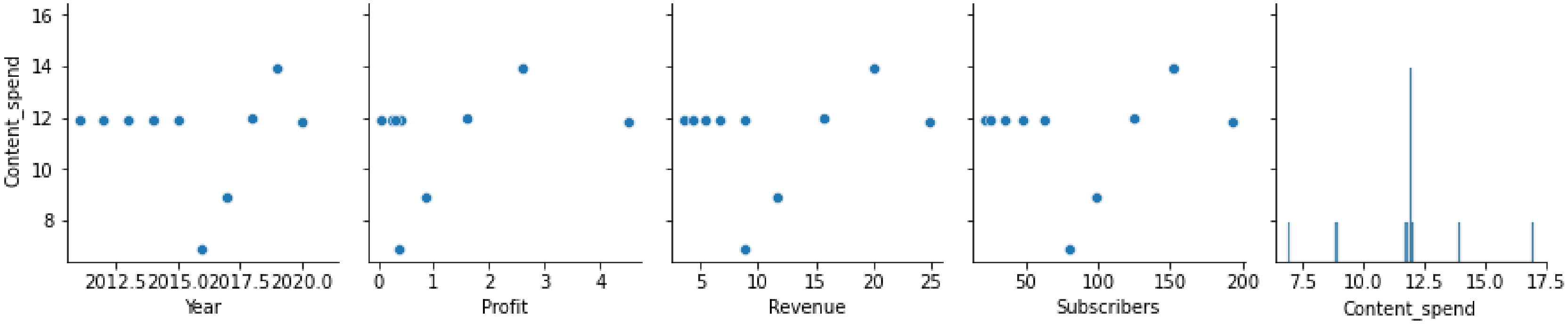
	Year	Profit	Revenue	Subscribers	Content_spend
0	2011	0.403	8.8	21.5	11.90
1	2012	0.050	3.5	25.7	11.90
2	2013	0.228	4.3	35.6	11.90
3	2014	0.403	5.4	47.9	11.90
4	2015	0.306	6.7	62.7	11.90
5	2016	0.379	8.8	79.9	6.88
6	2017	0.839	11.6	99.0	8.91
7	2018	1.600	15.7	124.3	12.00
8	2019	2.600	20.1	151.5	13.90
9	2020	4.500	24.9	192.9	11.80
10	2021	0.403	8.8	71.3	17.00

Understand the relationship between variables

In [18]:
```
sns.pairplot(df_new2)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x247c94faeb0>
```





In [19]:

```
df_new2.corr()
```

Out[19]:

	Year	Profit	Revenue	Subscribers	Content_spend
Year	1.000000	0.615799	0.701986	0.802945	0.337156
Profit	0.615799	1.000000	0.955805	0.924207	0.102073
Revenue	0.701986	0.955805	1.000000	0.955647	0.087678
Subscribers	0.802945	0.924207	0.955647	1.000000	0.027576
Content_spend	0.337156	0.102073	0.087678	0.027576	1.000000

Linear Regression Model

In [20]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing
from sklearn.metrics import r2_score
```

In [21]:

```
#split features: recognising dependent and independent variables
y=df_new2[['Revenue']]
print(y)
x=df_new2.drop(['Revenue'], axis=1)
print(x)
```

```
    Revenue
0       8.8
1       3.5
2       4.3
3       5.4
4       6.7
5       8.8
6      11.6
7      15.7
8      20.1
9      24.9
10      8.8
    Year  Profit  Subscribers  Content_spend
0   2011   0.403         21.5          11.90
1   2012   0.050         25.7          11.90
2   2013   0.228         35.6          11.90
3   2014   0.403         47.9          11.90
4   2015   0.306         62.7          11.90
5   2016   0.379         79.9           6.88
6   2017   0.839         99.0           8.91
7   2018   1.600        124.3          12.00
8   2019   2.600        151.5          13.90
9   2020   4.500        192.9          11.80
10  2021   0.403         71.3          17.00
```

In [22]:

```
#preparing training and test dataset
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
x_train.shape
print(x_train)
```

```
    Year  Profit  Subscribers  Content_spend
1   2012   0.050         25.7          11.90
8   2019   2.600        151.5          13.90
0   2011   0.403         21.5          11.90
7   2018   1.600        124.3          12.00
10  2021   0.403         71.3          17.00
3   2014   0.403         47.9          11.90
5   2016   0.379         79.9           6.88
4   2015   0.306         62.7          11.90
```

In [23]:

```
x_test
```

Out[23]:

	Year	Profit	Subscribers	Content_spend
2	2013	0.228	35.6	11.90
6	2017	0.839	99.0	8.91
9	2020	4.500	192.9	11.80

In [24]:

```
y_test
```

Out[24]:

	Revenue
2	4.3
6	11.6
9	24.9

In [25]:
```
y_train
```

Out[25]:

	Revenue
1	3.5
8	20.1
0	8.8
7	15.7
10	8.8
3	5.4
5	8.8
4	6.7

In [26]:
```
%%time
#instantiating linear regression model and fitting it to the training data
LR = LinearRegression()
LR.fit(x_train,y_train)
```

```
CPU times: total: 0 ns
Wall time: 7 ms
```

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

In [27]:
```
print('Intercept (c): ', LR.intercept_)
print('Coefficient (m): ', LR.coef_)
#scoring the model based on training and testing data
LR_test_score=LR.score(x_test, y_test)
LR_train_score=LR.score(x_train, y_train)
print('LR Testing Score: ', LR_test_score)
print('LR Trainig Score: ', LR_train_score)
```

```
Intercept (c):  [-1698.50826236]
Coefficient (m):  [[ 0.85049957  8.75480811 -0.08626368 -0.54048637]]
LR Testing Score:  0.4315848148229623
LR Trainig Score:  0.953214552157122
```

In [29]:

```
#predicting on test data
y_predict = LR.predict(x_test)
print(y_predict)
```

```
[[ 6.04070022]
 [10.93882338]
 [35.8795097 ]]
```

In [30]:

```
#evaluating Linear Regression model
MSE_LR=mean_squared_error(y_test,y_predict)
RMSE_LR=np.sqrt(mean_squared_error(y_test,y_predict))
print("LR mean_sqrd_error is==", MSE_LR)
print("LR root_mean_squared error of is==",RMSE_LR)

#r2 score
LR_score=r2_score(y_test, y_predict)
print('R2 Score:', LR_score)
```

```
LR mean_sqrd_error is== 41.338941700642025
LR root_mean_squared error of is== 6.429536662982957
R2 Score: 0.4315848148229623
```

In [31]:

```
#evaluating feature YEAR
x_yr=df_new2[['Year']]
y_yr=df_new2[['Revenue']]
x_yr_train, x_yr_test, y_yr_train, y_yr_test = train_test_split(x_yr, y_yr, test_size=0.25)
print('Shape of X_Year_TrainingData:', x_yr_train.shape)
LR_yr = LinearRegression()
LR_yr.fit(x_yr_train,y_yr_train)
y_yr_predict = LR_yr.predict(x_yr_test)
print(y_yr_predict)
print('Test Score:',LR_yr.score(x_yr_test, y_yr_test))
print('Train Score:', LR_yr.score(x_yr_train, y_yr_train))
print('Linear Model Coefficient (m): ', LR_yr.coef_)
print('Linear Model Coefficient (b): ', LR_yr.intercept_)
MSE_LR_yr=mean_squared_error(y_yr_test,y_yr_predict)
print("mean_sqrd_error is==", MSE_LR_yr)
LR_yr_score=r2_score(y_yr_test, y_yr_predict)
print('R2 Score:', LR_yr_score)
```

```
Shape of X_Year_TrainingData: (8, 1)
[[ 7.05217391]
 [10.675     ]
 [14.29782609]]
Test Score: 0.6622920541915764
Train Score: 0.40582987064824083
Linear Model Coefficient (m):  [[1.2076087]]
Linear Model Coefficient (b):  [-2423.86413043]
mean_sqrd_error is== 14.918436121612563
R2 Score: 0.6622920541915764
```

In [32]:

```
#evaluating feature CONTENT SPEND
x_cont=df_new2[['Content_spend']]
y_cont=df_new2[['Revenue']]
#print(x_sub)
#print(y_sub)
x_cont_train, x_cont_test, y_cont_train, y_cont_test = train_test_split(x_cont, y_cont, test_size=0.25)
#print('Shape of X_Subscribers_TrainingData:', x_sub_train.shape)
#print(x_sub_train)
LR_cont = LinearRegression()
LR_cont.fit(x_cont_train,y_cont_train)
y_cont_predict = LR_cont.predict(x_cont_test)
print(y_cont_predict)
print('Test Score:',LR_cont.score(x_cont_test, y_cont_test))
print('Train Score:', LR_cont.score(x_cont_train, y_cont_train))
print('Linear Model Coefficient (m): ', LR_cont.coef_)
print('Linear Model Coefficient (b): ', LR_cont.intercept_)
MSE_LR_cont=mean_squared_error(y_cont_test,y_cont_predict)
print("mean_sqrd_error is==", MSE_LR_cont)
LR_cont_score=r2_score(y_cont_test, y_cont_predict)
print('R2 Score:', LR_cont_score)
```

```
[[10.87815336]
 [12.84148935]
 [10.09594779]]
Test Score: -0.18828657615246636
Train Score: 0.0164665682114693
Linear Model Coefficient (m):  [[-0.39110279]]
Linear Model Coefficient (b):  [15.53227653]
mean_sqrd_error is== 56.950614639733885
R2 Score: -0.18828657615246636
```

In [33]:

```
#evaluating feature PROFIT
x_prof=df_new2[['Profit']]
y_prof=df_new2[['Revenue']]
#print(x_sub)
#print(y_sub)
x_prof_train, x_prof_test, y_prof_train, y_prof_test = train_test_split(x_prof, y_prof, test_size=0.25)
#print('Shape of X_Subscribers_TrainingData:', x_sub_train.shape)
#print(x_sub_train)
LR_prof = LinearRegression()
LR_prof.fit(x_prof_train,y_prof_train)
y_prof_predict = LR_prof.predict(x_prof_test)
print(y_prof_predict)
print('Test Score:',LR_prof.score(x_prof_test, y_prof_test))
print('Train Score:', LR_prof.score(x_prof_train, y_prof_train))
print('Linear Model Coefficient (m): ', LR_prof.coef_)
print('Linear Model Coefficient (b): ', LR_prof.intercept_)
MSE_LR_prof=mean_squared_error(y_prof_test,y_prof_predict)
print("mean_sqrd_error is==", MSE_LR_prof)
LR_prof_score=r2_score(y_prof_test, y_prof_predict)
print('R2 Score:', LR_prof_score)
```

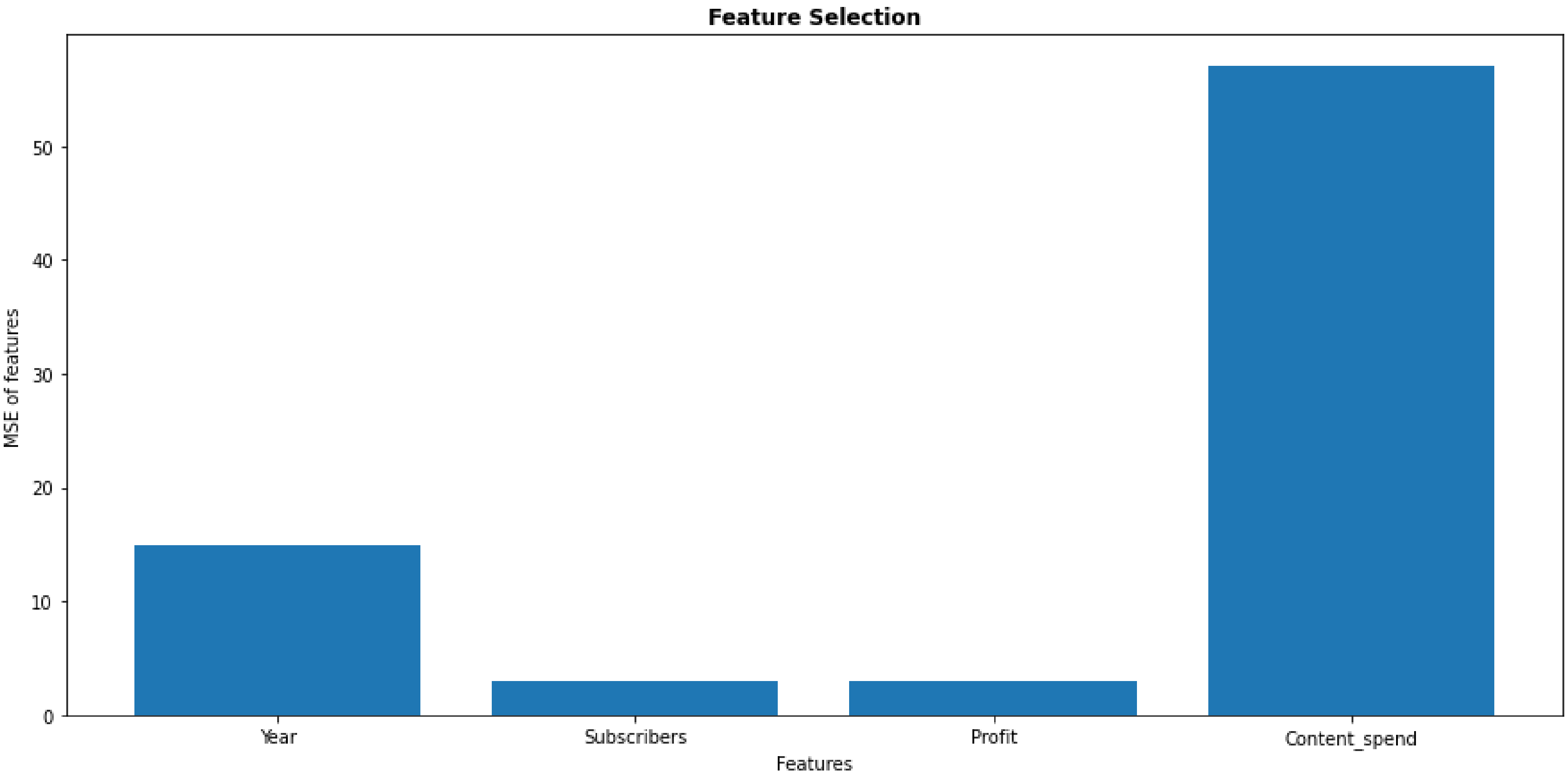
```
[[ 6.95308413]
 [ 7.39907115]
 [17.50044725]]
Test Score: 0.9154431651177327
Train Score: 0.9066228766199144
Linear Model Coefficient (m):  [[4.59780432]]
Linear Model Coefficient (b):  [5.54615601]
mean_sqrd_error is== 2.9281092399341575
R2 Score: 0.9154431651177327
```

In [34]:

```
#evaluating feature SUBSCRIBER
x_sub=df_new2[['Subscribers']]
y_sub=df_new2[['Revenue']]
#print(x_sub)
#print(y_sub)
x_sub_train, x_sub_test, y_sub_train, y_sub_test = train_test_split(x_sub, y_sub, test_size=0.25)
#print('Shape of X_Subscribers_TrainingData:', x_sub_train.shape)
#print(x_sub_train)
LR_sub = LinearRegression()
LR_sub.fit(x_sub_train,y_sub_train)
y_sub_predict = LR_sub.predict(x_sub_test)
print(y_sub_predict)
print('Test Score:',LR_sub.score(x_sub_test, y_sub_test))
print('Train Score:', LR_sub.score(x_sub_train, y_sub_train))
print('Linear Model Coefficient (m): ', LR_sub.coef_)
print('Linear Model Coefficient (b): ', LR_sub.intercept_)
MSE_LR_sub=mean_squared_error(y_sub_test,y_sub_predict)
print("mean_sqrd_error is==", MSE_LR_sub)
LR_sub_score=r2_score(y_sub_test, y_sub_predict)
print('R2 Score:', LR_sub_score)
```

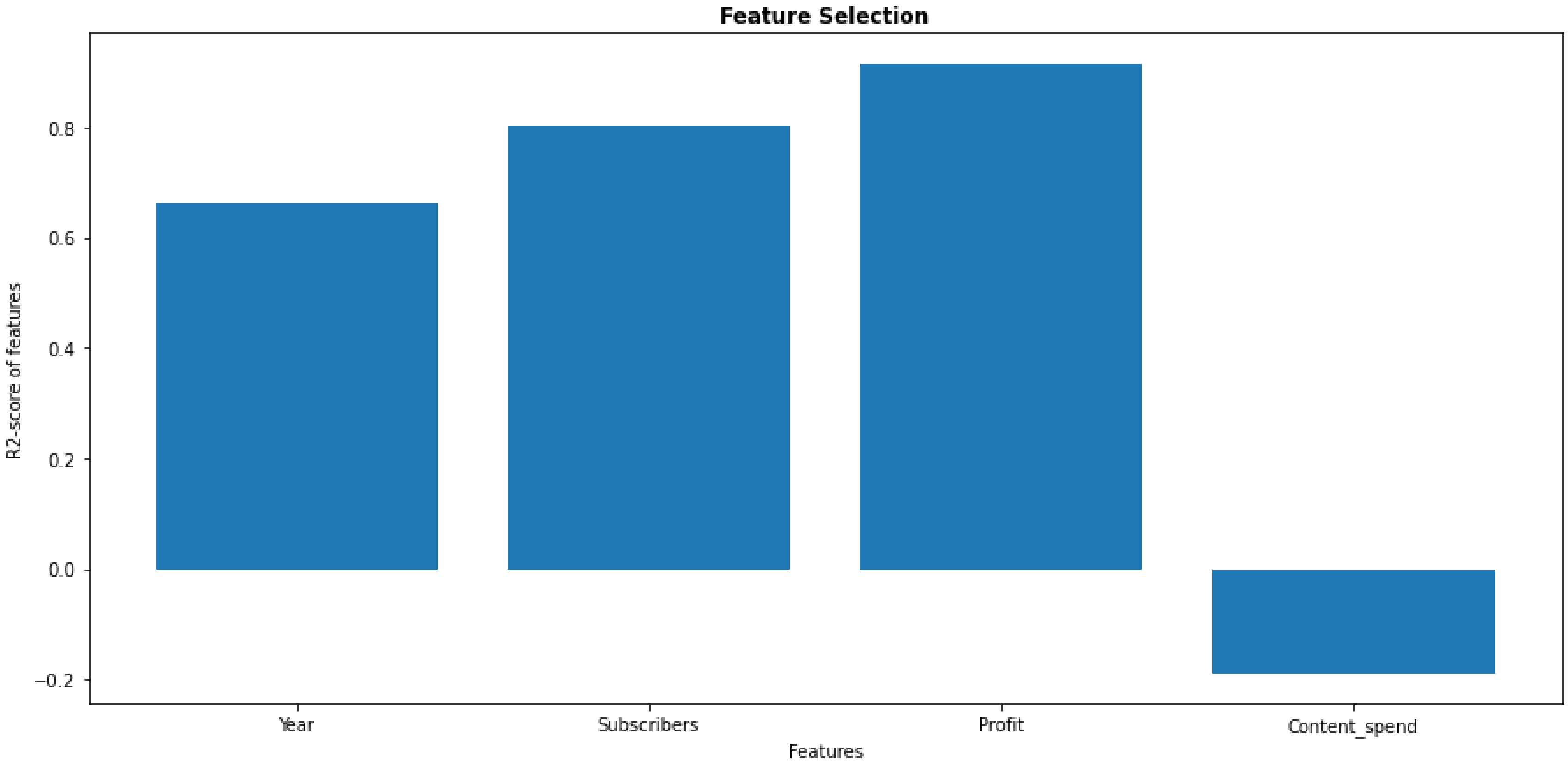
```
[[10.83355791]
 [16.07230472]
 [ 8.80413347]]
Test Score: 0.8037585420697845
Train Score: 0.9202296052045075
Linear Model Coefficient (m):  [[0.11798979]]
Linear Model Coefficient (b):  [1.40617345]
mean_sqrd_error is== 2.9004487482085843
R2 Score: 0.8037585420697845
```

In [35]:
```
plt.bar(["Year", "Subscribers","Profit","Content_spend"],[MSE_LR_yr,MSE_LR_sub,MSE_LR_prof,MSE_LR_cont])
plt.title("Feature Selection", fontweight='bold')
plt.ylabel("MSE of features")
plt.xlabel("Features")
plt.show()
```



In [36]:

```
plt.bar(["Year", "Subscribers","Profit","Content_spend"],[LR_yr_score,LR_sub_score,LR_prof_score,LR_cont_score])
plt.title("Feature Selection", fontweight='bold')
plt.ylabel("R2-score of features")
plt.xlabel("Features")
plt.show()
```



In [38]:

```
# predict revenue with #ofSubscribers
y_predict_1 = LR_sub.predict([[200]])
print('Revenue for 200 scubscribers:', y_predict_1)
# predict revenue for a particular year
y_predict_2 = LR_yr.predict([[2030]])
print('Revenue during year 2030:', y_predict_2)
```

```
Revenue for 200 scubscribers: [[25.00413204]]
Revenue during year 2030: [[27.58152174]]

C:\Users\vdp10002\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but LinearRegression was fitted with feature names
  warnings.warn(
C:\Users\vdp10002\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Random Forest Regression Model

In [39]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [40]:

```
%%time
#instantiating Random Forest regression model and fitting it to the training data
RF = RandomForestRegressor(n_jobs=-1)
RF.fit(x_sub_train, y_sub_train)
```

```
CPU times: total: 93.8 ms
Wall time: 87.4 ms

<timed exec>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
```

Out[40]:

```
RandomForestRegressor(n_jobs=-1)
```

In [41]:

```
#predicting on test data
y_predict_RF = RF.predict(x_sub_test)
print(y_predict_RF)
```

```
[ 9.229 12.973  7.821]
```

In [52]:
```
#evaluating Random Forest regression model
#score_RF=RF.score(y_sub_test,y_predict_RF)
MSE_RF=mean_squared_error(y_sub_test,y_predict_RF)
RMSE_RF=np.sqrt(mean_squared_error(y_sub_test,y_predict_RF))
RF_train_score=RF.score(x_sub_train, y_sub_train)
RF_test_score=RF.score(x_sub_test, y_sub_test)
print('RF Testing Score:',RF_test_score)
print('RF Training Score:',RF_train_score)
#print("RF r2 score is ",score_RF)
print("RF mean_sqrd_error is==", MSE_RF)
print("RF root_mean_squared error of is==",RMSE_RF)
RF_r2_score=r2_score(y_sub_test, y_predict_RF)
print('RF R2 Score:', RF_r2_score)
```

```
RF Testing Score: 0.7997922643211551
RF Training Score: 0.963208451309989
RF mean_sqrd_error is== 2.9590703333333264
RF root_mean_squared error of is== 1.720194853303929
RF R2 Score: 0.7997922643211551
```

In [58]:
```
# predict revenue with #ofSubscribers
y_predict_RF1 = RF.predict([[200]])
print('Revenue for 200 scubscribers:', y_predict_RF1)
```

```
Revenue for 200 scubscribers: [21.689]

C:\Users\vdp10002\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but RandomForestRegressor was fitted with feature names
  warnings.warn(
```

KNN Regression Model

In [44]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [45]:

```
%%time
#instantiating KNN regression model and fitting it to the training data
KNN = KNeighborsRegressor(n_neighbors=2)
KNN.fit(x_sub_train,y_sub_train)
```

```
CPU times: total: 0 ns
Wall time: 2 ms
```

Out[45]:

```
KNeighborsRegressor(n_neighbors=2)
```

In [46]:

```
y_predict_knn = KNN.predict(x_sub_test)
print(y_predict_knn)
```

```
[[10.2 ]
 [15.85]
 [ 7.1 ]]
```

In [53]:

```
KNN_test_score=KNN.score(x_sub_test, y_sub_test)
KNN_train_score=KNN.score(x_sub_train, y_sub_train)
print('KNN Testing Score:',KNN_test_score)
print('KNN Training Score:',KNN_train_score)
MSE_KNN=mean_squared_error(y_sub_test,y_predict_knn)
RMSE_KNN=np.sqrt(mean_squared_error(y_sub_test,y_predict_knn))
print("KNN MSE is ",MSE_KNN)
print("KNN RMSE is ",RMSE_KNN)
KNN_r2_score=r2_score(y_sub_test, y_predict_knn)
print('KNN R2 Score:', KNN_r2_score)
```

```
KNN Testing Score: 0.9516801984663961
KNN Training Score: 0.9262212823913122
KNN MSE is  0.7141666666666656
KNN RMSE is  0.8450838222724806
KNN R2 Score: 0.9516801984663961
```

In [61]:

```
# predict revenue with #ofSubscribers
y_predict_KNN1 = KNN.predict([[200]])
print('Revenue for 200 scubscribers:', y_predict_KNN1)
```

```
Revenue for 200 scubscribers: [[22.5]]

C:\Users\vdp10002\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but KNeighborsRegressor was fitted with feature names
  warnings.warn(
```

Decision Tree Regression Model

In [48]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [49]:

```
%%time
#instantiating Decision Tree regression model and fitting it to the training data
DT = DecisionTreeRegressor(random_state = 0)
DT.fit(x_sub_train,y_sub_train)
```

```
CPU times: total: 0 ns
Wall time: 2 ms
```

Out[49]:

```
DecisionTreeRegressor(random_state=0)
```

In [50]:

```
y_predict_DT = DT.predict(x_sub_test)
print(y_predict_DT)
```

```
[ 8.8 11.6  8.8]
```

In [54]:

```
DT_test_score=DT.score(x_sub_test, y_sub_test)
DT_train_score=DT.score(x_sub_train, y_sub_train)
print('DT Testing Score:',DT_test_score)
print('DT Training Score:',DT_train_score)
MSE_DT=mean_squared_error(y_sub_test,y_predict_DT)
RMSE_DT=np.sqrt(mean_squared_error(y_sub_test,y_predict_DT))
print("DTT MSE is ",MSE_DT)
print("DTT RMSE is ",RMSE_DT)
DT_r2_score=r2_score(y_sub_test, y_predict_DT)
print('DTT R2 Score:', DT_r2_score)
```

```
DT Testing Score: 0.5214253495714929
DT Training Score: 1.0
DTT MSE is  7.073333333333333
DTT RMSE is  2.6595739007091592
DTT R2 Score: 0.5214253495714929
```

In [60]:

```
# predict revenue with #ofSubscribers
y_predict_DT1 = DT.predict([[200]])
print('Revenue for 200 scubscribers:', y_predict_DT1)
```

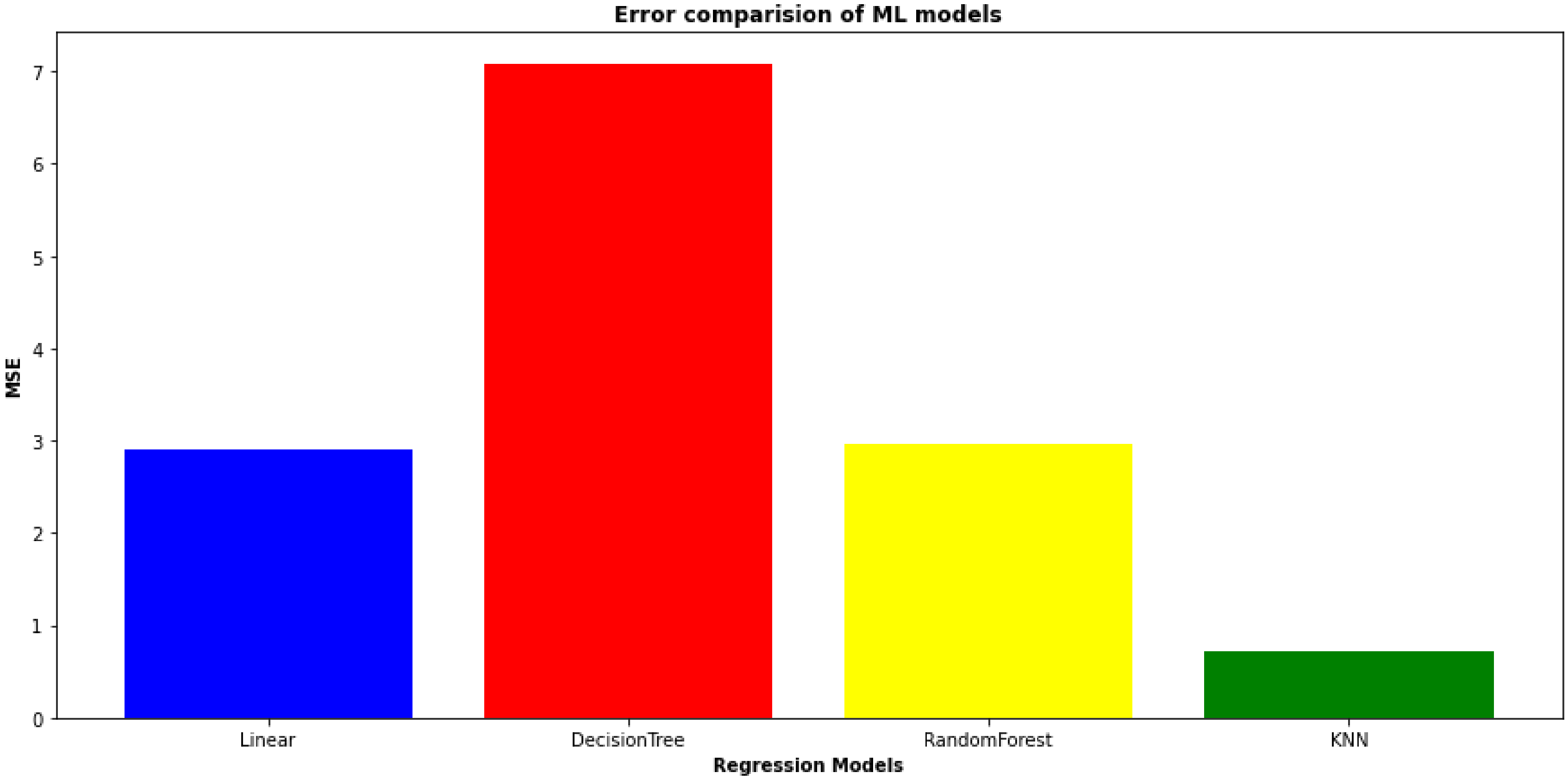
```
Revenue for 200 scubscribers: [24.9]
```

```
C:\Users\vdp10002\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature
names, but DecisionTreeRegressor was fitted with feature names
  warnings.warn(
```

Comparing Models

In [55]:
```
print(MSE_LR_sub)
print(MSE_DT)
print(MSE_RF)
print(MSE_KNN)
plt.bar(["Linear", "DecisionTree","RandomForest","KNN"],[MSE_LR_sub,MSE_DT,MSE_RF,MSE_KNN], color=['blue', 'red'
plt.title("Error comparision of ML models", fontweight="bold")
plt.ylabel("MSE", fontweight="bold")
plt.xlabel("Regression Models", fontweight="bold")
plt.show()
```

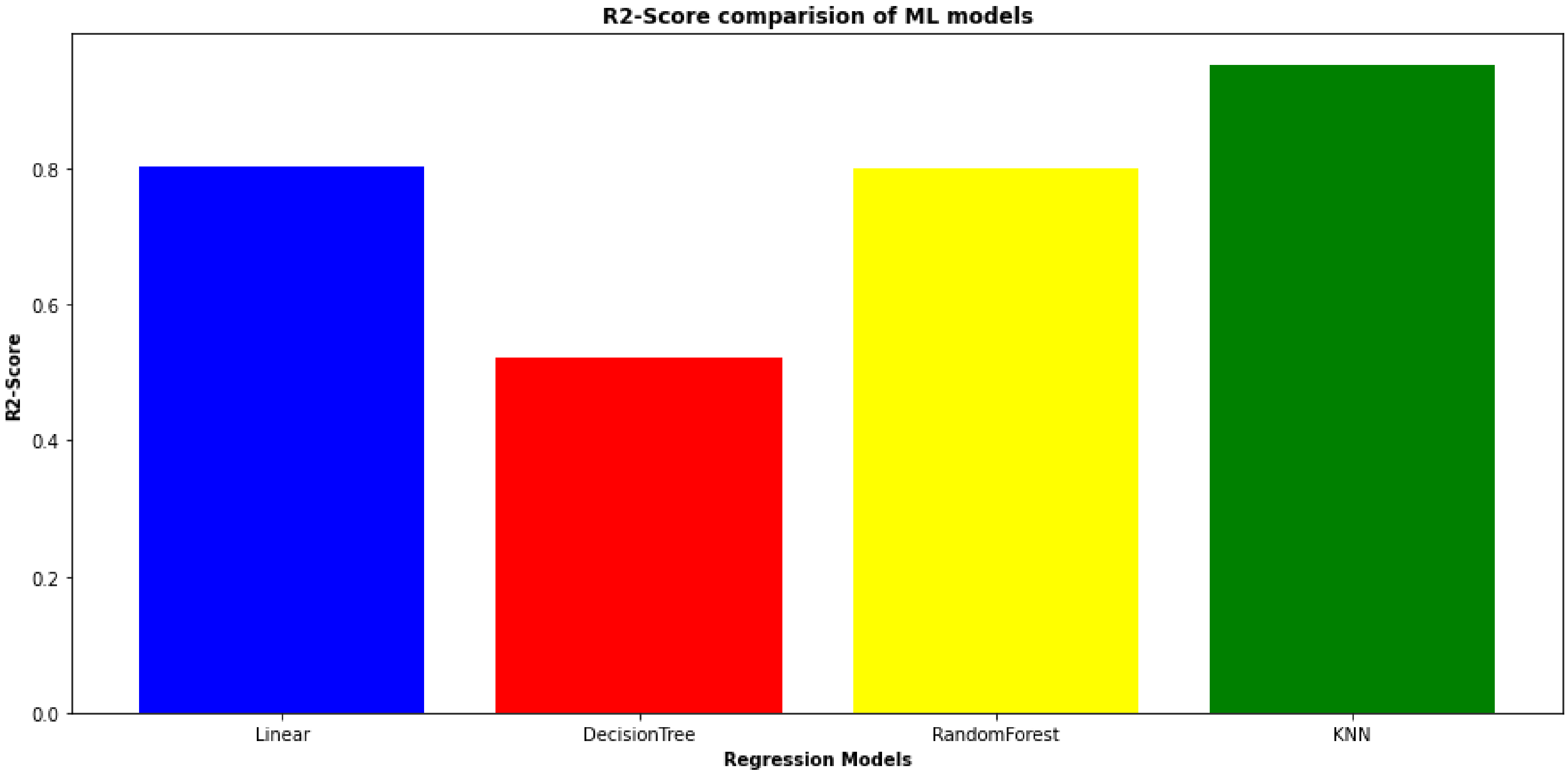
```
2.9004487482085843
7.073333333333333
2.959070333333264
0.7141666666666656
```



In [57]:

```
print(LR_sub_score)
print(DT_r2_score)
print(RF_r2_score)
print(KNN_r2_score)

plt.bar(["Linear", "DecisionTree","RandomForest","KNN"],[LR_sub_score,DT_r2_score, RF_r2_score,KNN_r2_score], co
plt.title("R2-Score comparision of ML models", fontweight="bold")
plt.ylabel("R2-Score", fontweight="bold")
plt.xlabel("Regression Models", fontweight="bold")
plt.show()
```

```
0.8037585420697845
0.5214253495714929
0.7997922643211551
0.9516801984663961
```



In [ ]:

```

```