In [162…

```python
import pandas as pd
import numpy as np
import datetime
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler
```

In [163…

```python
movie_data=pd.read_csv(r"C:\Users\Ziad Ahmed\Desktop\Faculty\AI\tmdb-movies (2).csv")
```

In [163…

```python
movie_data.head()
```

Out[163…

	id	imdb_id	popularity	original_title	cast	homepage	director	tagline	keywords	overview	runtime	genres	production
0	135397	tt0369610	32.985763	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	Colin Trevorrow	The park is open.	monster|dna|tyrannosaurus rex|velociraptor|island	Twenty-two years after the events of Jurassic ...	124	Action|Adventure|Science Fiction|Thriller	Universal Stu Entertainme
1	76341	tt1392190	28.419936	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	George Miller	What a Lovely Day.	future|chase|post-apocalyptic|dystopia|australia	An apocalyptic story set in the furthest reach...	120	Action|Adventure|Science Fiction|Thriller	Villag Pictures|Ke
2	262500	tt2908446	13.112507	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Robert Schwentke	One Choice Can Destroy You	based on novel|revolution|dystopia|sequel|dyst...	Beatrice Prior must confront her inner demons ...	119	Adventure|Science Fiction|Thriller	Entertainmen Films
3	140607	tt2488496	11.173104	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	J.J. Abrams	Every generation has a story.	android|spaceship|jedi|space opera|3d	Thirty years after defeating the Galactic Empi...	136	Action|Adventure|Science Fiction|Fantasy	Lucasfil Production
4	168259	tt2820852	9.335014	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	James Wan	Vengeance Hits Home	car race|speed|revenge|suspense|car	Deckard Shaw seeks revenge against Dominic Tor...	137	Action|Crime|Thriller	Pict Film|Me

In [163…

```python
movie_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   original_title        10866 non-null  object 
 4   cast                  10790 non-null  object 
 5   homepage              2936 non-null   object 
 6   director              10822 non-null  object 
 7   tagline               8042 non-null   object 
 8   keywords              9373 non-null   object 
 9   overview              10862 non-null  object 
 10  runtime               10866 non-null  int64  
 11  genres                10843 non-null  object 
 12  production_companies  9836 non-null   object 
 13  release_date          10866 non-null  object 
 14  vote_count            10866 non-null  int64  
 15  vote_average          10866 non-null  float64
 16  release_year          10866 non-null  int64  
 17  budget_adj            10866 non-null  float64
 18  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(4), object(11)
memory usage: 1.6+ MB
```

In [163…

```python
movie_data.isnull().sum()
```

Out[163…

```
id                         0
imdb_id                   10
popularity                 0
original_title             0
cast                      76
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

In [163…

```python
#removing  row duplicates from the data frame.
movie_data.drop_duplicates(inplace = True)
movie_data.shape
```

Out[163…

```
(10865, 19)
```

In [163…

```python
print("Rows with zero values in the budget_adj column:",movie_data[(movie_data['budget_adj']==0)].shape[0])
print("Rows with zero values in the revenue_adj column:",movie_data[(movie_data['revenue_adj']==0)].shape[0])
```

```
Rows with zero values in the budget_adj column: 5696
Rows with zero values in the revenue_adj column: 6016
```

In [163…

```python
#Replace un necessary rows with the mean value of each column
movie_data['budget_adj'] = movie_data['budget_adj'].replace(0, movie_data['budget_adj'].mean())
movie_data['revenue_adj'] = movie_data['revenue_adj'].replace(0, movie_data['revenue_adj'].mean())
```

In [163…

```python
sr=pd.Series(['production_companies','tagline']).is_unique
print(sr)
```

```
True
```

In [163…

```python
#Removing un necessary columns from the data frame
movie_data.drop(['id','imdb_id','homepage','director','tagline','release_date','production_companies'],axis=1,inplace=True)
movie_data.shape
```

Out[163…

```
(10865, 12)
```

In [163…

```python
movie_data.isnull().sum()
```

Out[163…

```
popularity           0
original_title       0
cast                76
keywords          1493
overview             4
runtime              0
genres              23
vote_count           0
vote_average         0
release_year         0
budget_adj           0
revenue_adj          0
dtype: int64
```

In [164…

```python
movie_data.shape
```

Out[164…

```
(10865, 12)
```

In [164…

```python
#Dealing with NaN values with proper imputation techniques

movie_data['overview'].fillna(movie_data['overview'].mode()[0], inplace=True)

movie_data['cast'].fillna(movie_data['cast'].mode()[0], inplace=True)

movie_data['keywords'].fillna(movie_data['keywords'].mode()[0], inplace=True)

movie_data['genres'].fillna(movie_data['genres'].mode()[0], inplace=True)
```

In [164…

```python
movie_data.isnull().sum()
```

Out[164…

```
popularity        0
original_title    0
cast              0
keywords          0
overview          0
runtime           0
genres            0
vote_count        0
vote_average      0
release_year      0
budget_adj        0
revenue_adj       0
dtype: int64
```

In [164…

```python
movie_data.head()
```

Out[164…

	popularity	original_title	cast	keywords	overview	runtime	genres	vote_count	vote_average	release_year	budget_adj	revenue_adj
0	32.985763	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	monster|dna|tyrannosaurus rex|velociraptor|island	Twenty-two years after the events of Jurassic ...	124	Action|Adventure|Science Fiction|Thriller	5562	6.5	2015	137999939.3	1.392446e+09
1	28.419936	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	future|chase|post-apocalyptic|dystopia|australia	An apocalyptic story set in the furthest reach...	120	Action|Adventure|Science Fiction|Thriller	6185	7.1	2015	137999939.3	3.481613e+08
2	13.112507	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	based on novel|revolution|dystopia|sequel|dyst...	Beatrice Prior must confront her inner demons ...	119	Adventure|Science Fiction|Thriller	2480	6.3	2015	101199955.5	2.716190e+08
3	11.173104	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	android|spaceship|jedi|space opera|3d	Thirty years after defeating the Galactic Empi...	136	Action|Adventure|Science Fiction|Fantasy	5292	7.5	2015	183999919.0	1.902723e+09
4	9.335014	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	car race|speed|revenge|suspense|car	Deckard Shaw seeks revenge against Dominic Tor...	137	Action|Crime|Thriller	2947	7.3	2015	174799923.1	1.385749e+09

In [164…

```python
#Convert the used categorical columns to numerical columns using label encoding techniques
label_encoder = preprocessing.LabelEncoder()
movie_data['original_title']= label_encoder.fit_transform(movie_data['original_title'])
movie_data['cast']= label_encoder.fit_transform(movie_data['cast'])
movie_data['overview']= label_encoder.fit_transform(movie_data['overview'])
movie_data['keywords']= label_encoder.fit_transform(movie_data['keywords'])
```

In [164…

```python
##Convert the used categorical columns to numerical columns using One hot encoding
movie_data['genres']=movie_data['genres'].str.get_dummies(sep='|')
movie_data.head()
```

Out[164…

	popularity	original_title	cast	keywords	overview	runtime	genres	vote_count	vote_average	release_year	budget_adj	revenue_adj
0	32.985763	4423	1739	5064	9631	124	1	5562	6.5	2015	137999939.3	1.392446e+09
1	28.419936	5061	10084	3100	2942	120	1	6185	7.1	2015	137999939.3	3.481613e+08
2	13.112507	4149	9278	695	3577	119	0	2480	6.3	2015	101199955.5	2.716190e+08
3	11.173104	7375	3697	292	9254	136	1	5292	7.5	2015	183999919.0	1.902723e+09
4	9.335014	3210	10331	1320	4196	137	1	2947	7.3	2015	174799923.1	1.385749e+09

In [ ]:

In [164…

```python
#the net profit which is the difference between (revenue_adj - budget_adj).
movie_data['netprofit']=movie_data['revenue_adj']-movie_data['budget_adj']
movie_data.head()
```

Out[164…

	popularity	original_title	cast	keywords	overview	runtime	genres	vote_count	vote_average	release_year	budget_adj	revenue_adj	netprofit
0	32.985763	4423	1739	5064	9631	124	1	5562	6.5	2015	137999939.3	1.392446e+09	1.254446e+09
1	28.419936	5061	10084	3100	2942	120	1	6185	7.1	2015	137999939.3	3.481613e+08	2.101614e+08
2	13.112507	4149	9278	695	3577	119	0	2480	6.3	2015	101199955.5	2.716190e+08	1.704191e+08
3	11.173104	7375	3697	292	9254	136	1	5292	7.5	2015	183999919.0	1.902723e+09	1.718723e+09
4	9.335014	3210	10331	1320	4196	137	1	2947	7.3	2015	174799923.1	1.385749e+09	1.210949e+09

In [164…

```python
X = movie_data.drop(['budget_adj','revenue_adj','netprofit'],axis=1)
Y = movie_data['netprofit']
```

In [164…

```python
print(movie_data[(movie_data['netprofit']==0)].shape[0])
```

```
4
```

In [164…

```python
#movie_data['netprofit'] = movie_data['netprofit'].replace(0, movie_data['netprofit'].mean())
```

In [165…

```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
X.head()
```

```
(10865, 10) (8692, 10) (2173, 10)
```

Out[165…

	popularity	original_title	cast	keywords	overview	runtime	genres	vote_count	vote_average	release_year
0	32.985763	4423	1739	5064	9631	124	1	5562	6.5	2015
1	28.419936	5061	10084	3100	2942	120	1	6185	7.1	2015
2	13.112507	4149	9278	695	3577	119	0	2480	6.3	2015
3	11.173104	7375	3697	292	9254	136	1	5292	7.5	2015
4	9.335014	3210	10331	1320	4196	137	1	2947	7.3	2015

In [165…

```python
#movie_data=movie_data.astype(int)
```

In [165…

```python
movie_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10865 entries, 0 to 10865
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      10865 non-null  float64
 1   original_title  10865 non-null  int32  
 2   cast            10865 non-null  int32  
 3   keywords        10865 non-null  int32  
 4   overview        10865 non-null  int32  
 5   runtime         10865 non-null  int64  
 6   genres          10865 non-null  int64  
 7   vote_count      10865 non-null  int64  
 8   vote_average    10865 non-null  float64
 9   release_year    10865 non-null  int64  
 10  budget_adj      10865 non-null  float64
 11  revenue_adj     10865 non-null  float64
 12  netprofit       10865 non-null  float64
dtypes: float64(5), int32(4), int64(4)
memory usage: 1018.6 KB
```

In [165…

```python
#apply feature scaling (normalization) for variables
scaler=StandardScaler()
X=scaler.fit_transform(X)
```

In [165…

```python
#Aplly Linear Regression Model
lin_reg_model = LinearRegression()
lin_reg_model.fit(X_train,Y_train)
```

Out[165…

```
LinearRegression()
```

In [165…

```python
training_data_prediction = lin_reg_model.predict(X_train)
error_score = mean_squared_error(Y_train, training_data_prediction)
print("Mean squared training Error : ", error_score)
```

```
Mean squared training Error :  8746267041966680.0
```

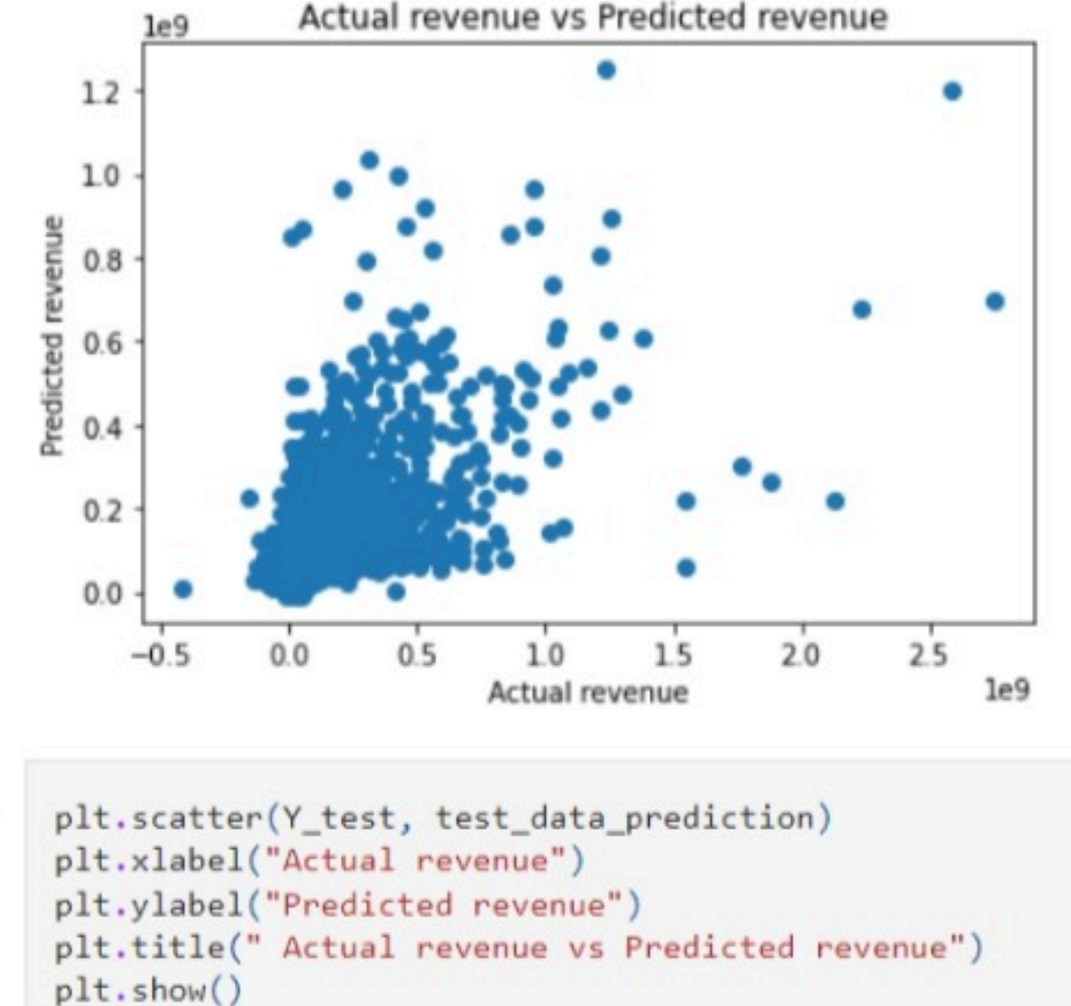
In [165…

```python
test_data_prediction = lin_reg_model.predict(X_test)
error_score = mean_squared_error(Y_test, test_data_prediction)
print("Mean squared testing Error : ", error_score)
```

```
Mean squared testing Error :  6952690476719321.0
```

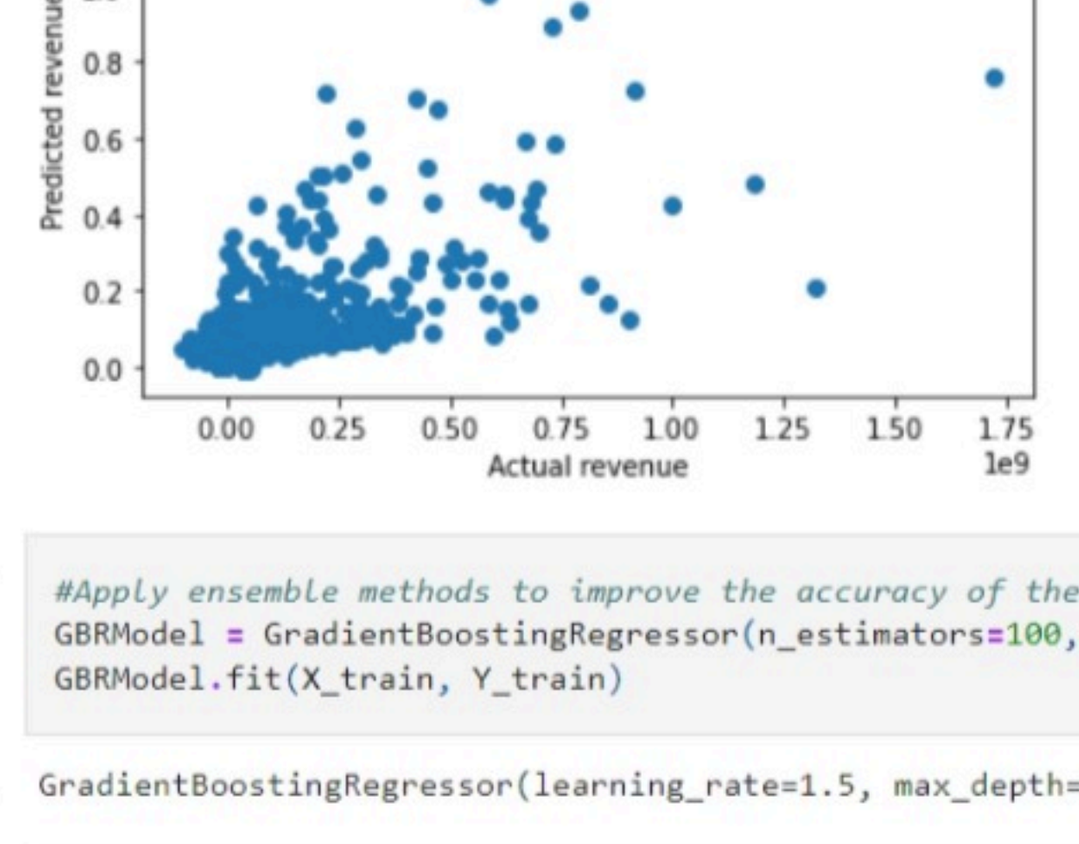
In [165…

```python
plt.scatter(Y_train, training_data_prediction)
plt.xlabel("Actual revenue")
plt.ylabel("Predicted revenue")
plt.title(" Actual revenue vs Predicted revenue")
plt.show()
```



In [165…

```python
plt.scatter(Y_test, test_data_prediction)
plt.xlabel("Actual revenue")
plt.ylabel("Predicted revenue")
plt.title(" Actual revenue vs Predicted revenue")
plt.show()
```



In [165…

```python
#Apply ensemble methods to improve the accuracy of the model
GBRModel = GradientBoostingRegressor(n_estimators=100,max_depth=5,learning_rate = 1.5 ,random_state=33)
GBRModel.fit(X_train, Y_train)
```

Out[165…

```
GradientBoostingRegressor(learning_rate=1.5, max_depth=5, random_state=33)
```

In [166…

```python
print('GBRModel Train Score is : ' , GBRModel.score(X_train, Y_train))
```

```
GBRModel Train Score is :  0.9729413427629218
```

In [166…

```python
print('GBRModel Test Score is : ' , GBRModel.score(X_test, Y_test))
```

```
GBRModel Test Score is :  -0.6233332275085979
```

In [166…

```python
y_train_pred = GBRModel.predict(X_train)
y_test_pred = GBRModel.predict(X_test)
```

In [166…

```python
MSEValue = mean_squared_error(Y_train, y_train_pred)
print('Mean Squared Train Error Value is : ', MSEValue)
```

```
Mean Squared Train Error Value is :  406250352682186.9
```

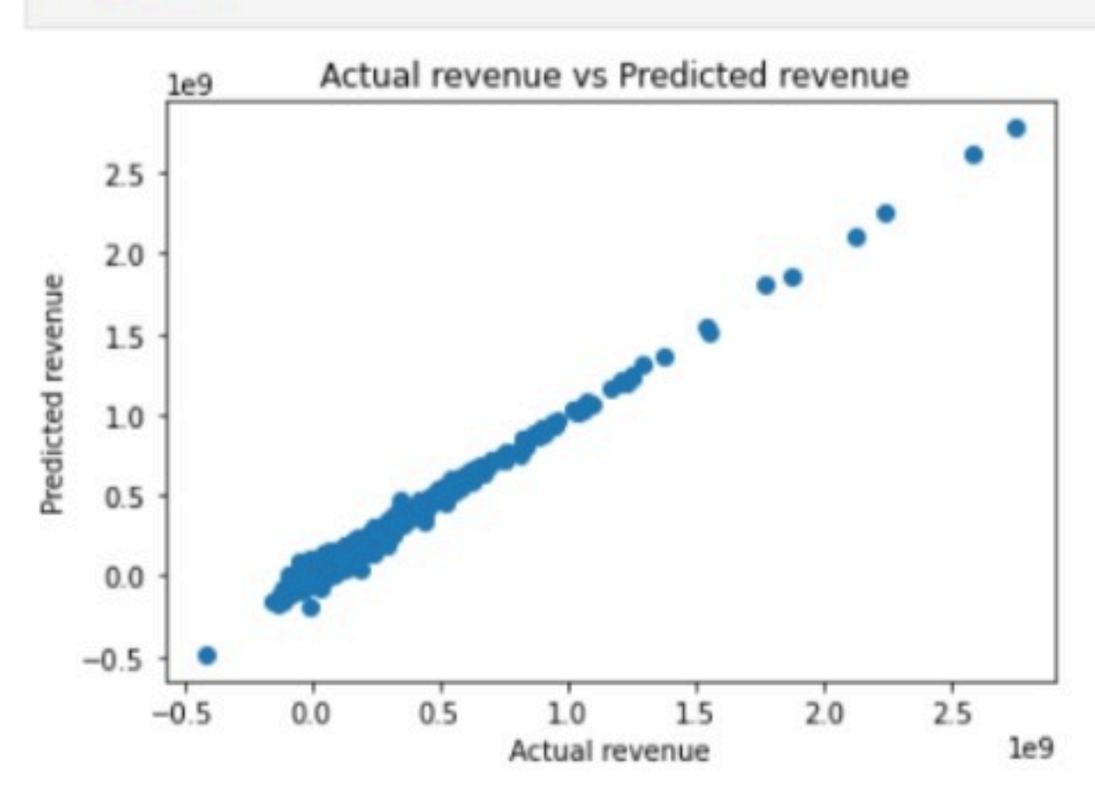
In [166…

```python
MSEValue = mean_squared_error(Y_test, y_pred)
print('Mean Squared  Test Error Value is : ', MSEValue)
```

```
Mean Squared  Test Error Value is :  1.5767954909925756e+16
```

In [166…

```python
plt.scatter(Y_train, y_train_pred)
plt.xlabel("Actual revenue")
plt.ylabel("Predicted revenue")
plt.title(" Actual revenue vs Predicted revenue")
plt.show()
```



In [166…

```python
plt.scatter(Y_test, y_pred)
plt.xlabel("Actual revenue")
plt.ylabel("Predicted revenue")
plt.title(" Actual revenue vs Predicted revenue")
plt.show()
```