

ABOUT PRICE OF CINEMA TICKETS PROJECT

objective: About movie ticket pricing system using python and machine learning By using kaggle source we uploaded cinemaTickets_Ref.csv

By importing these libraries we can done these project

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

To read the given data

****EDA**

In [3]:

```
sc = pd.read_csv('cinemaTicket_Ref.csv')
```

To print five five rows

In [4]:

```
sc.head()
```

Out[4]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_p
0	1492	304	3900000	26	0	4	4.26	1500
1	1492	352	3360000	42	0	5	8.08	800
2	1492	489	2560000	32	0	4	20.00	800
3	1492	429	1200000	12	0	1	11.01	1000
4	1492	524	1200000	15	0	3	16.67	800

To print last five rows

In [5]:

```
sc.tail()
```

Out[5]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ti
142519	1569	495	1320000	22	0	2	3.86	
142520	1569	474	1200000	15	0	1	65.22	
142521	1569	524	1060000	8	0	3	9.20	
142522	1569	529	600000	5	0	2	5.00	
142523	1569	486	250000	5	0	1	1.79	



To print the size of given data

In [6]:

```
sc.shape
```

Out[6]:

(142524, 14)

To show is there any null values present because to understand is there any need to do data cleaning

In [7]:

```
sc.isnull()
```

Out[7]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ti
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	
142519	False	False	False	False	False	False	False	
142520	False	False	False	False	False	False	False	
142521	False	False	False	False	False	False	False	
142522	False	False	False	False	False	False	False	
142523	False	False	False	False	False	False	False	

142524 rows × 14 columns



To show the total sum of null values present in given data

In [8]:

```
sc.isnull().sum()
```

Out[8]:

film_code	0
cinema_code	0
total_sales	0
tickets_sold	0
tickets_out	0
show_time	0
occu_perc	125
ticket_price	0
ticket_use	0
capacity	125
date	0
month	0
quarter	0
day	0
dtype:	int64

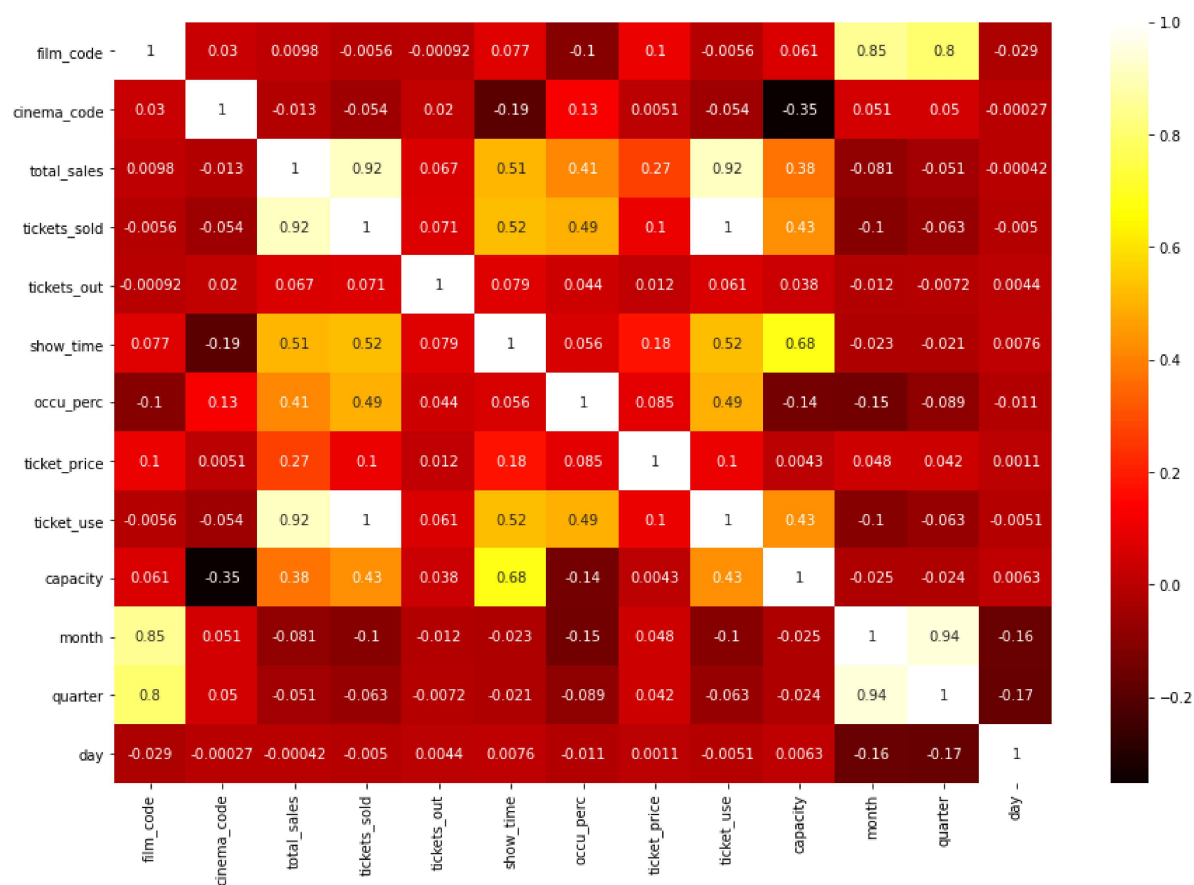
In []:

In [9]:

```
plt.figure(figsize=(15, 10))
sns.heatmap(sc.corr(), annot=True, cmap='hot')
```

Out[9]:

<matplotlib.axes._subplots.AxesSubplot at 0x1a791e16790>

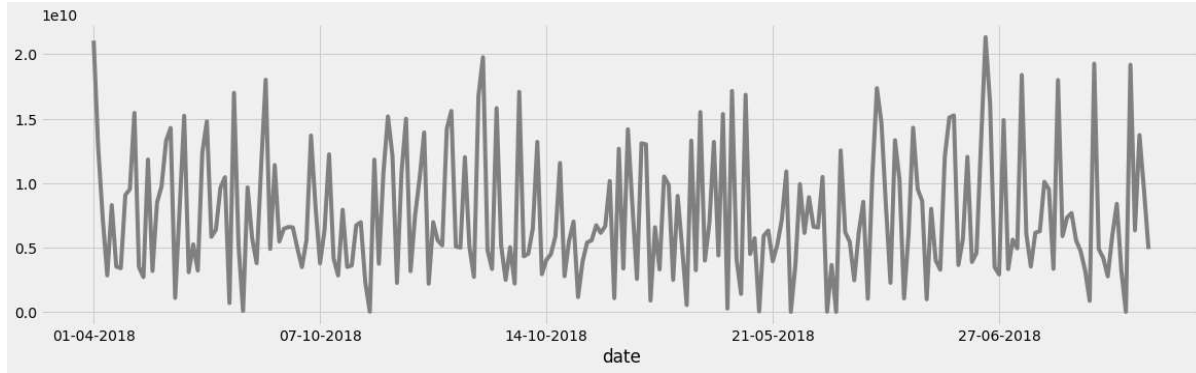


In [17]:

```
plt.style.use('fivethirtyeight')
sc.groupby('date')['total_sales'].sum().plot(figsize=(18, 5), color='grey')
```

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x1a7aeb6f790>



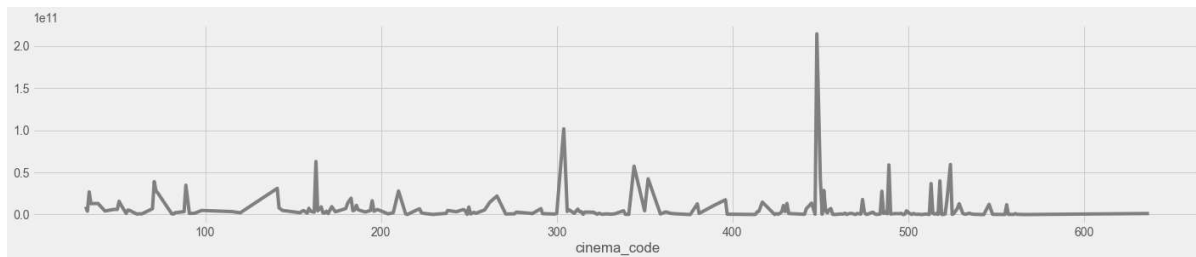
TOTAL SALE PER DAY

In [123]:

```
plt.style.use('fivethirtyeight')
sc.groupby('cinema_code')['total_sales'].sum().plot(figsize=(22, 4), color='grey')
```

Out[123]:

<matplotlib.axes._subplots.AxesSubplot at 0x26716b66400>



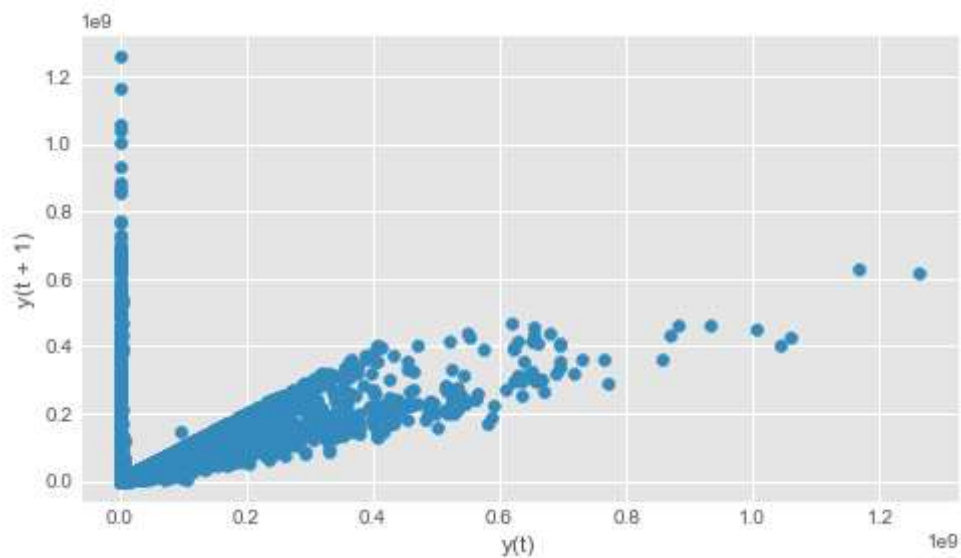
SCATTER PLOT

In [124]:

```
plt.style.use('ggplot')
plt.figure(figsize=(7, 4))
pd.plotting.lag_plot(sc['total_sales'], lag=1)
```

Out[124]:

<matplotlib.axes._subplots.AxesSubplot at 0x26712075a90>



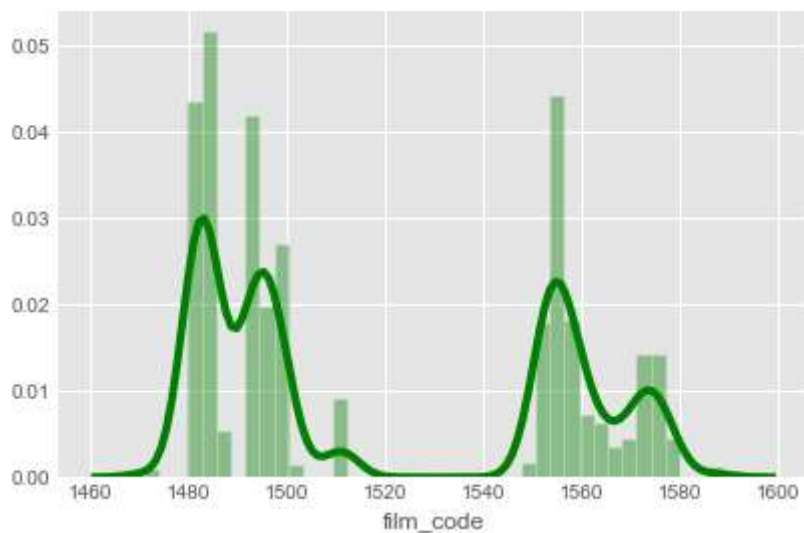
DIST PLOT

In [125]:

```
sns.distplot(sc['film_code'],color='Green',bins=40)
```

Out[125]:

<matplotlib.axes._subplots.AxesSubplot at 0x2671205bc40>



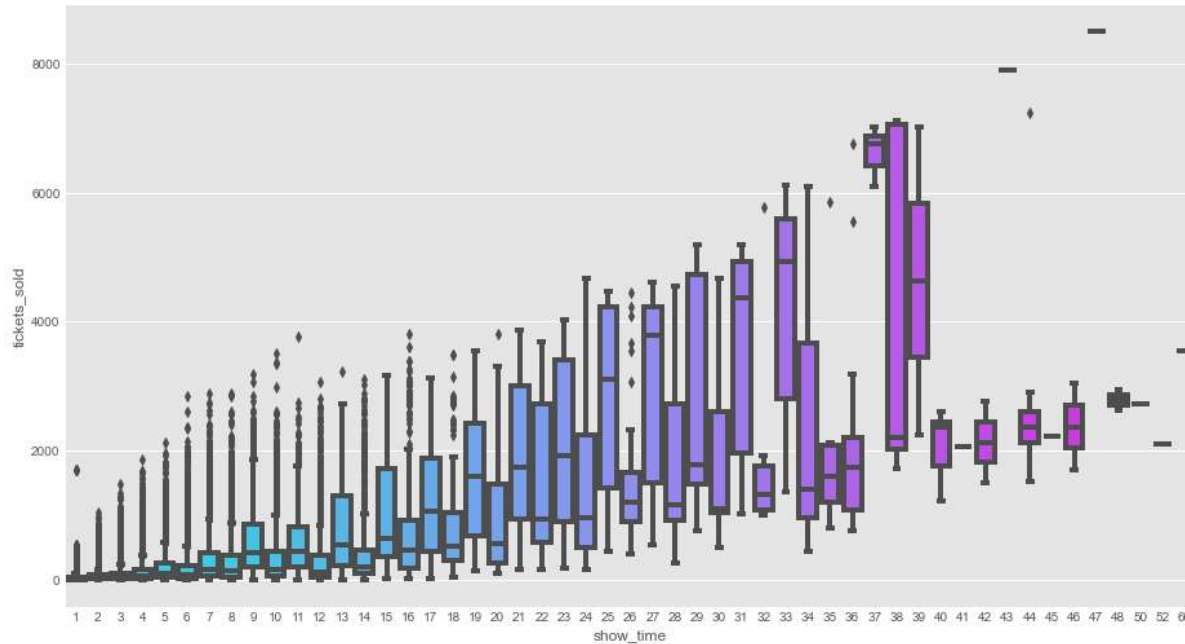
BOX PLOT

In [126]:

```
plt.figure(figsize=(14,8))  
sns.boxplot(x='show_time',y='tickets_sold',data=sc,palette='cool')
```

Out[126]:

<matplotlib.axes._subplots.AxesSubplot at 0x26712195160>



DATA CLEANING

In [128]:

```
# to delete the rows having null values
sc.dropna()
# to delete the date column
sc.drop('date', inplace=True, axis=1)
print(sc)
```

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	\
0	1492	304	3900000	26	0	
1	1492	352	3360000	42	0	
2	1492	489	2560000	32	0	
3	1492	429	1200000	12	0	
4	1492	524	1200000	15	0	
...
142519	1569	495	1320000	22	0	
142520	1569	474	1200000	15	0	
142521	1569	524	1060000	8	0	
142522	1569	529	600000	5	0	
142523	1569	486	250000	5	0	

	show_time	occu_perc	ticket_price	ticket_use	capacity	month	\
0	4	4.26	150000.0	26	610.328639	5	
1	5	8.08	80000.0	42	519.801980	5	
2	4	20.00	80000.0	32	160.000000	5	
3	1	11.01	100000.0	12	108.991826	5	
4	3	16.67	80000.0	15	89.982004	5	
...
142519	2	3.86	60000.0	22	569.948187	11	
142520	1	65.22	80000.0	15	22.999080	11	
142521	3	9.20	132500.0	8	86.956522	11	
142522	2	5.00	120000.0	5	100.000000	11	
142523	1	1.79	50000.0	5	279.329609	11	

	quarter	day
0	2	5
1	2	5
2	2	5
3	2	5
4	2	5
...
142519	4	4
142520	4	4
142521	4	4
142522	4	4
142523	4	4

[142524 rows x 13 columns]

To delete the columns having null values

In [130]:

```
sc = sc.dropna(axis=1)
print(sc)
```

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	\
0	1492	304	3900000	26	0	
1	1492	352	3360000	42	0	
2	1492	489	2560000	32	0	
3	1492	429	1200000	12	0	
4	1492	524	1200000	15	0	
...
142519	1569	495	1320000	22	0	
142520	1569	474	1200000	15	0	
142521	1569	524	1060000	8	0	
142522	1569	529	600000	5	0	
142523	1569	486	250000	5	0	

	show_time	ticket_price	ticket_use	month	quarter	day
0	4	150000.0	26	5	2	5
1	5	80000.0	42	5	2	5
2	4	80000.0	32	5	2	5
3	1	100000.0	12	5	2	5
4	3	80000.0	15	5	2	5
...
142519	2	60000.0	22	11	4	4
142520	1	80000.0	15	11	4	4
142521	3	132500.0	8	11	4	4
142522	2	120000.0	5	11	4	4
142523	1	50000.0	5	11	4	4

[142524 rows x 11 columns]

To check whether null values are deleted or not

In [131]:

```
sc.isnull().sum()
```

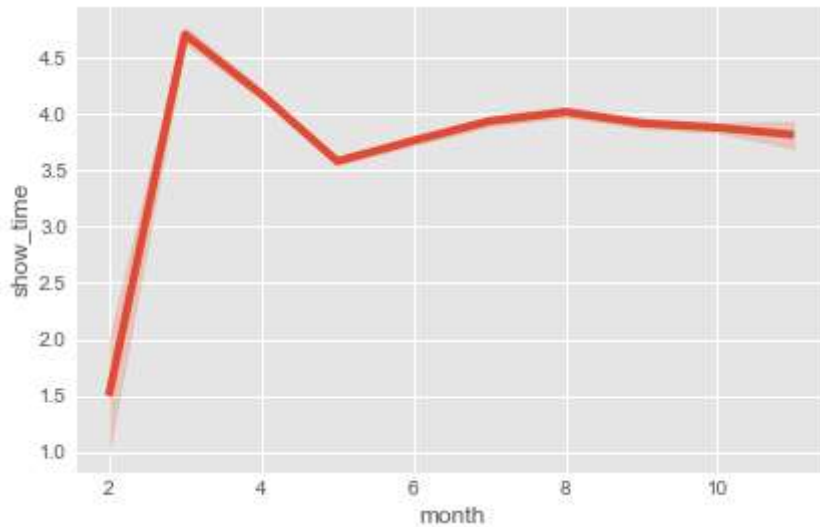
Out[131]:

```
film_code      0
cinema_code    0
total_sales    0
tickets_sold   0
tickets_out    0
show_time      0
ticket_price   0
ticket_use     0
month          0
quarter        0
day            0
dtype: int64
```

lineplot

In [76]:

```
sns.lineplot(data=sc,x="month",y="show_time")  
plt.show()
```



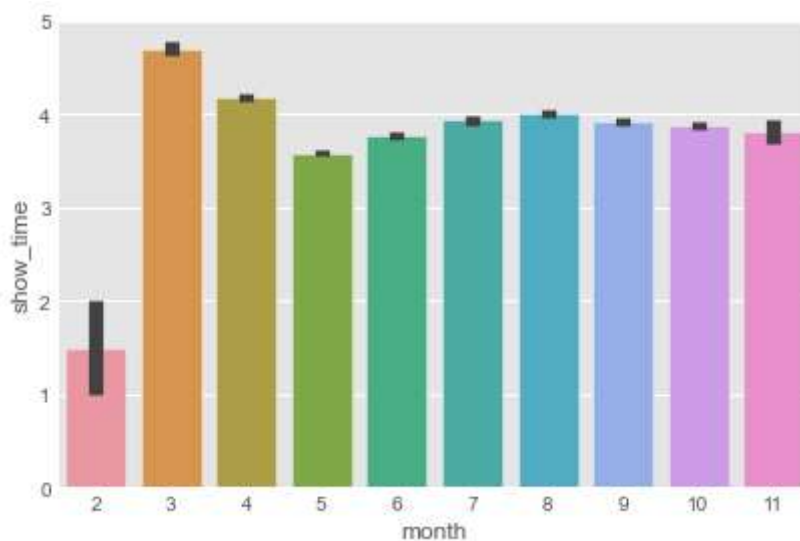
BAR PLOT

In [133]:

```
sns.barplot(data=sc,x="month",y="show_time")
```

Out[133]:

<matplotlib.axes._subplots.AxesSubplot at 0x26716c55b20>



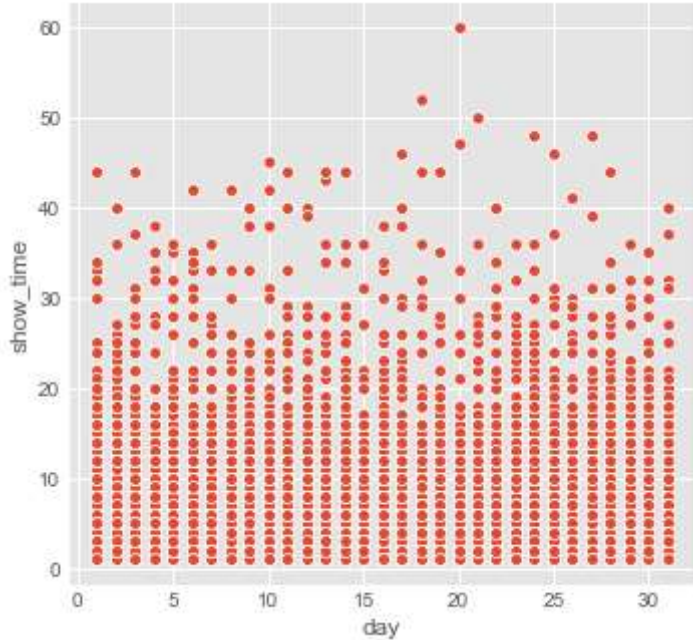
scatter plot

In [135]:

```
plt.figure(figsize=(5,5))
sns.scatterplot(x='day',y='show_time',data=sc,palette='rocket_r')
```

Out[135]:

<matplotlib.axes._subplots.AxesSubplot at 0x26712080340>



****First machine learning model 1.K NEAREST NEIGHBOUR**

In [136]:

```
import sklearn
#features
X = sc.iloc[:, [2,3,6,7]].values
#target
y = sc.iloc[:, 4].values
```

In [137]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

****splitting our data into 80 to 20 percentage**

In [138]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
```

In [139]:

```
from sklearn.preprocessing import StandardScaler
s = StandardScaler()
X_train = s.fit_transform(X_train)
X_test = s.transform(X_test)
```

****FITTING**

In [140]:

```
#FITTING
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

Out[140]:

KNeighborsClassifier()

****PREDICTION**

In [141]:

```
y_pred = classifier.predict(X_test)
```

In [142]:

```
y_test
```

Out[142]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [143]:

```
y_pred
```

Out[143]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

****ACCURACY**

In [144]:

```

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
print(cm)
print("accuracy=", ac)

```

```

[[27276      0      1 ...      0      0      0]
 [  283      5      1 ...      0      0      0]
 [  319      7     17 ...      0      0      0]
 ...
 [      0      0      0 ...      0      0      0]
 [      1      0      0 ...      0      0      0]
 [      0      0      0 ...      0      0      0]]
accuracy= 0.9579372039992984

```

****second machine learning model 2.NAIVE BAYES**

****selecting features and target**

In [145]:

```

X = sc.iloc[:, [2, 3, 6, 7]].values
y = sc.iloc[:, 4].values

```

In [146]:

```

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:, 0] = le.fit_transform(X[:, 0])

```

****splitting our data**

In [147]:

```

#splitting our data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

```

In [148]:

```

from sklearn.preprocessing import StandardScaler
s = StandardScaler()
X_train = s.fit_transform(X_train)
X_test = s.transform(X_test)

```

****fitting**

In [149]:

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

Out[149]:

GaussianNB()

**prediction

In [150]:

```
y_pred = classifier.predict(X_test)
```

In [151]:

```
y_pred
```

Out[151]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [152]:

```
y_test
```

Out[152]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [166]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
print(cm)
```

```
[[26418    0    26 ...    0    0    0]
 [   260    0     1 ...    0    0    0]
 [   314    0     0 ...    0    0    0]
 ...
 [     1    0     0 ...    0    0    0]
 [     0    0     0 ...    0    0    0]
 [     0    0     0 ...    0    0    0]]
```

**accuracy

In [167]:

```
print("accuracy=", ac)
```

accuracy= 0.9269251008594983

****Third machine learning model**

****3.LOGISTIC REGRESSION**

In []:

```
from sklearn.linear_model import LogisticRegression
#splitting our data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state =
0)
# Define model. Specify a number for random_state to ensure same results each run
sc_model = LogisticRegression(random_state=1)
# Fit model
X_train = sc.iloc[:,[2,7]]
Y_train = sc.iloc[:,4]
sc_model.fit(X_train,Y_train)
```

prediction and accuracy

In [169]:

```
LogisticRegression_score = sc_model.score(X_train,Y_train)
print('Accuracy obtained by LogisticRegression_sc_model',LogisticRegression_score*100)
```

Accuracy obtained by LogisticRegression_sc_model 95.6821307288597

summary: Cinema industry is not excluded of getting advantage of predictive modeling. Like other industry e.g. retail , banking and restaurants , sale forecast can help cinemas for cost reduction and better ROI. By forecasting sale, screening in different location could be optimized as well as effective market targeting and pricing. Also historical data of sale and movies details e.g. cost, cast and crews, and other project details like schedule, could help producers to select high performance cast and crews and planning for better projects ROI . Also it helps to assign screening location on hot spots and areas. **1.EDA**

In statistics, exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed.

2.DATA CLEANING

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

3.splitting our data into 80 percent for training and 20 percent for testing

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset. Train Dataset: Used to fit the machine learning model. Test Dataset: Used to evaluate the fit machine learning model.

4.BUILDING THREE MACHINE LEARNING MODELS

1.K nearest neighbour The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slower as the size of that data in use grows.

2.Naive bayes It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. 3.Logistic Regression Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).