Classification

-Prepare the model :

-We are about to build our prediction model… 😊

-we cleaned the data and visualized it.

-To build a classification model we need to import (Decision Tree Classifier) from “scikit learn”.

-Before classification we need to transform our nonnumeric data columns into numeric ones, to do that we import (LabelEncoder) from “scikit learn” package .

```
37]: from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     clm_txt=['original_title','cast','director','genres','release_date','production_companies']
     for i in clm_txt:
         movie[i]=le.fit_transform(movie[i])
```

```
38]: movie.head(2)
```

38]:

	New_ID	popularity	budget	revenue	original_title	cast	director	runtime	genres	production_companies	release_
112	0	1.41239	5000000	54418872	898	1228	509	103	98	129	
113	1	1.40805	700000	17986781	404	1461	868	103	242	368	

-After transforming the type of the nonnumeric columns we have to select the “Target column”

- In this dataset we selected “vote_average” to be the target column.

- We converted its values into (0,1) .

```
In [277]: def vote_average_app(x):
              if x>=5 : return 1
              if x<5 : return 0
```

```
In [278]: movie['vote_average']=movie['vote_average'].apply(vote_average_app)
          movie['vote_average'].value_counts()
```

```
Out[278]: 1    1525
          0     135
          Name: vote_average, dtype: int64
```

-After that we split the data into (x_train , y_train , x_test , y_test) then we used the classifier to fit(x_train , y_train) and predict if the vote_average=1 or 0.

```
In [227]: from sklearn import tree
          from sklearn.model_selection import train_test_split
          X = np.array(movie.drop(columns=['vote_average']))
          Y = np.array(movie['vote_average'])
          X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size=0.3, random_state=36)
          train_test_split(Y, shuffle=False)
          clf = tree.DecisionTreeClassifier()
          clf = clf.fit(X_train,y_train)
```

```
In [228]: y_pred=clf.predict(X_test)
          y_pred
```

-we have prediction model with accuracy 86%.

```
In [281]: from sklearn import metrics
          metrics.accuracy_score(y_test,y_pred)
```

```
Out[281]: 0.8634538152610441
```

-we visualized the relation between the values of the actual column and the values of the predicted column.

[232]:
```
from sklearn.metrics import confusion_matrix
co_matrix=confusion_matrix(y_test,y_pred)
co_matrix
```

[232]:
```
array([[ 13,  38],
       [ 27, 420]], dtype=int64)
```

[233]:
```
sns.heatmap(co_matrix,center=True,annot=True,fmt='d')
plt.show()
```