In [5]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score

df=pd.read_csv("Top_Highest_Openings.csv")

# Step 1: Handle Missing Values
  # Drop rows with missing values, or use df.fillna() to impute missing values

# Step 2: Handle Duplicates
df = df.drop_duplicates()  # Remove duplicate rows if they exist
df.head()
```

Out[5]:

	Release	Opening	Total Gross	% of Total	Theaters	Average	Date	Distributor
0	Avengers: Endgame	357115007	858373000	41.6	4662.0	76601	26-04-2019	Walt Disney Studios Motion Pictures
1	Spider-Man: No Way Home	260138569	804793477	32.3	4336.0	59995	17-12-2021	Sony Pictures Releasing
2	Avengers: Infinity War	257698183	678815482	38.0	4474.0	57599	27-04-2018	Walt Disney Studios Motion Pictures
3	Star Wars: Episode VII - The Force Awakens	247966675	936662225	26.5	4134.0	59982	NaN	Walt Disney Studios Motion Pictures
4	Star Wars: Episode VIII - The Last Jedi	220009584	620181382	35.5	4232.0	51987	15-12-2017	Walt Disney Studios Motion Pictures

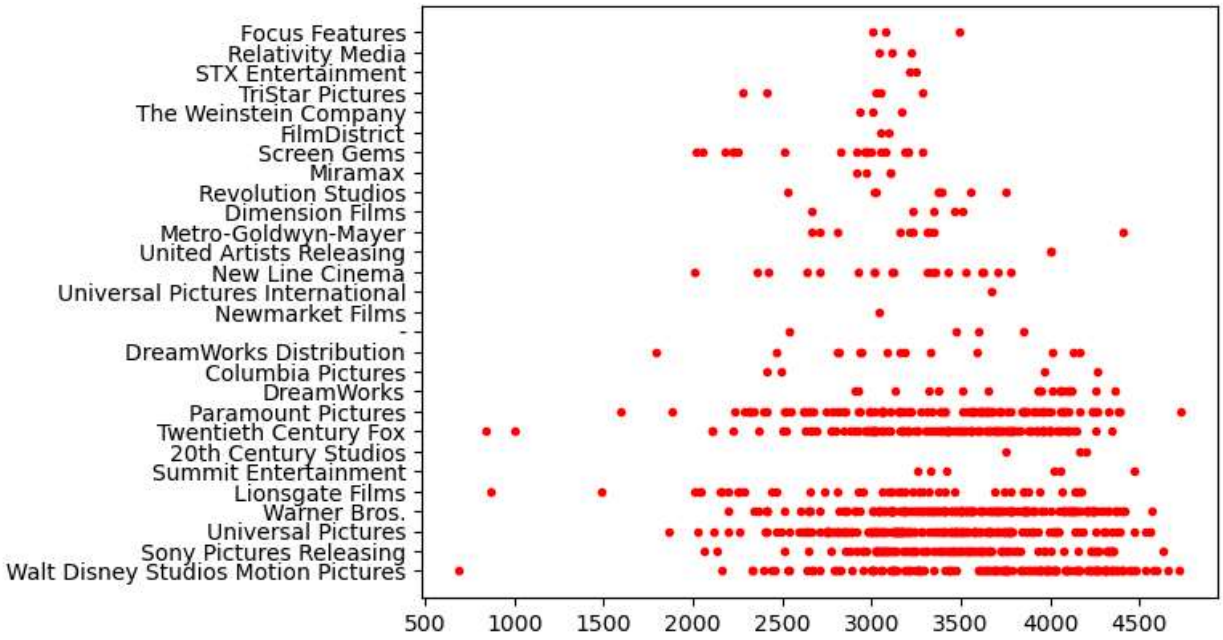
In [6]:
```
df.drop(['Release','Total Gross','% of Total','Average'],axis='columns',inplace=True)
df.head()
```

Out[6]:

	Opening	Theaters	Date	Distributor
0	357115007	4662.0	26-04-2019	Walt Disney Studios Motion Pictures
1	260138569	4336.0	17-12-2021	Sony Pictures Releasing
2	257698183	4474.0	27-04-2018	Walt Disney Studios Motion Pictures
3	247966675	4134.0	NaN	Walt Disney Studios Motion Pictures
4	220009584	4232.0	15-12-2017	Walt Disney Studios Motion Pictures

In [7]:
```
from matplotlib import pyplot as plt
plt.scatter(df.Theaters,df.Distributor,marker='.',color='red')
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x22363226590>
```



In [8]:
```
round((df.isnull().sum()/df.shape[0])*100,2)
```

Out[8]:
```
Opening         0.0
Theaters        5.3
Date           29.5
Distributor     0.0
dtype: float64
```

In [9]:
```
df = df.drop(columns='Date')
df = df.fillna(df.Theaters.mean())

df.isnull().sum()
```

Out[9]:
```
Opening        0
Theaters       0
Distributor    0
dtype: int64
```

In [10]:
```
round((df.isnull().sum()/df.shape[0])*100,2)
```

Out[10]:
```
Opening        0.0
Theaters       0.0
Distributor    0.0
dtype: float64
```

In [119…
```
#Convert Distributor column
dummies = pd.get_dummies(df['Distributor'])

df = df.drop('Distributor',axis=1)
df.head(5)

df = pd.concat([df, dummies], axis=1)
df.head()
```

Out[119]:

	Opening	Theaters	-	20th Century Studios	Columbia Pictures	Dimension Films	DreamWorks	DreamWorks Distribution	FilmDistric
0	357115007	4662.0	False	False	False	False	False	False	Fals
1	260138569	4336.0	False	False	False	False	False	False	Fals
2	257698183	4474.0	False	False	False	False	False	False	Fals
3	247966675	4134.0	False	False	False	False	False	False	Fals
4	220009584	4232.0	False	False	False	False	False	False	Fals

5 rows × 30 columns

In [104…

```
from sklearn.preprocessing import MinMaxScaler

# Normalize the dataset
scaler_trainx = MinMaxScaler()
scaler_trainy = MinMaxScaler()
scaler_testx = MinMaxScaler()
scaler_testy = MinMaxScaler()
```

In [105…

```
#'target' is the column you want to predict
target = df['Opening']
df = df.drop('Opening', axis=1)

#X = scaled_df.drop('Opening', axis=1)


# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size=0.2, random_

X_train =scaler_trainx.fit_transform(pd.DataFrame(X_train))
y_train =scaler_trainy.fit_transform(pd.DataFrame(y_train))
X_test =scaler_testx.fit_transform(pd.DataFrame(X_test))
y_test =scaler_testy.fit_transform(pd.DataFrame(y_test))
```

In [106…

```
from sklearn.tree import DecisionTreeRegressor



# Creating a Decision Tree regression model with max_depth=10 and min_samples_split=5
model = DecisionTreeRegressor(max_depth=100, min_samples_split=6, random_state=42)

# Training the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
#print("Mean Squared Error:", mse)
```

```
print(mse)
print("Prediction before denormalization:", y_pred)
print("Predictions after denromalization:", scaler_testy.inverse_transform([y_pred]))
```

```
0.028228573203616148
Prediction before denormalization: [0.01659087 0.00527422 0.00422445 0.07014637 0.001
09828 0.01705397
 0.0898538  0.25039342 0.02654728 0.14492467 0.00535207 0.09478907
 0.05305068 0.02654728 0.00527422 0.00149055 0.01500954 0.05142355
 0.02503182 0.01500954 0.15953746 0.05082983 0.07776565 0.03708391
 0.017223   0.06576642 0.06576642 0.05165616 0.01018614 0.02403259
 0.08786523 0.22198537 0.03765757 0.04954141 0.00526186 0.25280307
 0.02997319 0.12956533 0.15178567 0.25177792 0.49344009 0.05054092
 0.07193404 0.01355193 0.05902637 0.03626505 0.01101538 0.22198537
 0.02429214 0.13704435 0.02697802 0.0240746  0.00526186 0.00872254
 0.40422127 0.07329786 0.0210119  0.25280307 0.08254608 0.03528748
 0.02302214 0.07329786 0.00149055 0.03051555 0.08254608 0.00526186
 0.03626505 0.05098459 0.08208903 0.02368773 0.02287808 0.02287808
 0.02403259 0.01101538 0.17486144 0.10263454 0.00872254 0.3216127
 0.011027   0.11832209 0.05328812 0.25861437 0.14190994 0.13149374
 0.1368533  0.07943431 0.01142589 0.03626505 0.01086466 0.04388479
 0.0210119  0.02997319 0.02287808 0.01239377 0.0571152  0.0571152
 0.0571152  0.25592412 0.08795746 0.03051555 0.02503182 0.35299866
 0.05662942 0.49344009 0.02287808 0.01018614 0.03638581 0.12956533
 0.14648425 0.04658612 0.04226676 0.00422445 0.01919063 0.02302214
 0.04825692 0.04317722 0.49344009 0.04226676 0.03177509 0.08387404
 0.00535207 0.017223   0.0181831  0.11208012 0.05165616 0.02289178
 0.00993583 0.00580152 0.03511089 0.02289178 0.02038417 0.07219672
 0.08407787 0.01919063 0.02628602 0.11003956 0.25592412 0.0240746
 0.31635673 0.22198537 0.07546121 0.00109828 0.02503182 0.02287808
 0.02429214 0.17485042 0.02997319 0.017223   0.04324839 0.03570852
 0.05165616 0.1171934  0.04770961 0.02953644 0.02355421 0.17485042
 0.11612365 0.00780099 0.13149374 0.0407064  0.02075156 0.07943431
 0.00527422 0.03051555 0.00697569 0.1171934  0.03511089 0.01081846
 0.01292008 0.07219672 0.02287808 0.00461942 0.08181495 0.03626505
 0.09478907 0.03618419 0.02313196 0.03638581 0.03765757 0.04607338
 0.5061265  0.01519407 0.03400703 0.01500954 0.00872254 0.37734094
 0.04231749 0.07005892 0.01018614 0.02287808 0.03051555 0.02061213
 0.00993583 0.22198537 0.04411782 0.0240746  0.11208012 0.017223
 0.00149055 0.3216127 ]
Predictions after denromalization: [[2.41089893e+07 2.22296641e+07 2.20553321e+07 3.3
0028012e+07
  2.15361773e+07 2.41858946e+07 3.62755592e+07 6.29359250e+07
  2.57624216e+07 4.54210236e+07 2.22425915e+07 3.70951458e+07
  3.01637675e+07 2.57624216e+07 2.22296641e+07 2.16013197e+07
  2.38463818e+07 2.98935549e+07 2.55107529e+07 2.38463818e+07
  4.78477291e+07 2.97949563e+07 3.42681133e+07 2.75122098e+07
  2.42139644e+07 3.22754352e+07 3.22754352e+07 2.99321840e+07
  2.30453728e+07 2.53448137e+07 3.59453218e+07 5.82182804e+07
  2.76074772e+07 2.95809928e+07 2.22276110e+07 6.33360887e+07
  2.63313537e+07 4.28703410e+07 4.65604117e+07 6.31658450e+07
  1.03298007e+08 2.97469790e+07 3.32996748e+07 2.36043210e+07
  3.11561339e+07 2.73762244e+07 2.31830822e+07 5.82182804e+07
  2.53879176e+07 4.41123617e+07 2.58339528e+07 2.53517913e+07
  2.22276110e+07 2.28023166e+07 8.84816875e+07 3.35261598e+07
  2.48431769e+07 6.33360887e+07 3.50619860e+07 2.72138818e+07
  2.51770113e+07 3.35261598e+07 2.16013197e+07 2.64214219e+07
  3.50619860e+07 2.22276110e+07 2.73762244e+07 2.98206572e+07
  3.49860857e+07 2.52875449e+07 2.51530874e+07 2.51530874e+07
  2.53448137e+07 2.31830822e+07 5.03925408e+07 3.83980199e+07
  2.28023166e+07 7.47631126e+07 2.31850122e+07 4.10032079e+07
  3.02031991e+07 6.43011557e+07 4.49203771e+07 4.31905866e+07
  4.40806343e+07 3.45452234e+07 2.32512549e+07 2.73762244e+07
  2.31580523e+07 2.86416133e+07 2.48431769e+07 2.63313537e+07
```

```
  2.51530874e+07 2.34119880e+07 3.08387509e+07 3.08387509e+07
  3.08387509e+07 6.38543938e+07 3.59606395e+07 2.64214219e+07
  2.55107529e+07 7.99752912e+07 3.07580792e+07 1.03298007e+08
  2.51530874e+07 2.30453728e+07 2.73962796e+07 4.28703410e+07
  4.56800190e+07 2.90902164e+07 2.83729119e+07 2.20553321e+07
  2.45407239e+07 2.51770113e+07 2.93676814e+07 2.85241095e+07
  1.03298007e+08 2.83729119e+07 2.66305892e+07 3.52825165e+07
  2.22425915e+07 2.42139644e+07 2.43734060e+07 3.99666213e+07
  2.99321840e+07 2.51553623e+07 2.30038046e+07 2.23172316e+07
  2.71845573e+07 2.51553623e+07 2.47389315e+07 3.33432973e+07
  3.53163660e+07 2.45407239e+07 2.57190342e+07 3.96277524e+07
  6.38543938e+07 2.53517913e+07 7.38902675e+07 5.82182804e+07
  3.38854220e+07 2.15361773e+07 2.55107529e+07 2.51530874e+07
  2.53879176e+07 5.03907110e+07 2.63313537e+07 2.42139644e+07
  2.85359277e+07 2.72838027e+07 2.99321840e+07 4.08157704e+07
  2.92767904e+07 2.62588234e+07 2.52653705e+07 5.03907110e+07
  4.06381196e+07 2.26492771e+07 4.31905866e+07 2.81137879e+07
  2.47999421e+07 3.45452234e+07 2.22296641e+07 2.64214219e+07
  2.25122227e+07 4.08157704e+07 2.71845573e+07 2.31503796e+07
  2.34993913e+07 3.33432973e+07 2.51530874e+07 2.21209238e+07
  3.49405689e+07 2.73762244e+07 3.70951458e+07 2.73627963e+07
  2.51952489e+07 2.73962796e+07 2.76074772e+07 2.90050672e+07
  1.05404804e+08 2.38770258e+07 2.70012407e+07 2.38463818e+07
  2.28023166e+07 8.40177450e+07 2.83813369e+07 3.29882778e+07
  2.30453728e+07 2.51530874e+07 2.64214219e+07 2.47767885e+07
  2.30038046e+07 5.82182804e+07 2.86803118e+07 2.53517913e+07
  3.99666213e+07 2.42139644e+07 2.16013197e+07 7.47631126e+07]]
```

In [107…

```
from sklearn.metrics import accuracy_score

model.score(X_test,y_test)
```

Out[107]:

```
0.06454059947267854
```

In [108…

```
from sklearn.linear_model import LinearRegression


# Creating a Linear Regression model
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
#print("Prediction before denormalization:", y_pred)


data = pd.DataFrame({
    'Theaters': [4200],
    '-': 0,
    '20th Century Studios': 0,
    'Columbia Pictures': 0,
    'Dimension Films': 0,
    'DreamWorks': 0,
```

```
     'DreamWorks Distribution': 0,
     'FilmDistrict': 0,
     'Focus Features': 0,
     'Lionsgate Films': 0,
     'Metro-Goldwyn-Mayer': 0,
     'Miramax': 0,
     'New Line Cinema': 0,
     'Newmarket Films': 0,
     'Paramount Pictures': 0,
     'Relativity Media': 0,
     'Revolution Studios': 0,
     'STX Entertainment': 0,
     'Screen Gems': 0,
     'Sony Pictures Releasing': 0,
     'Summit Entertainment': 0,
     'The Weinstein Company': 0,
     'TriStar Pictures': 0,
     'Twentieth Century Fox': 0,
     'United Artists Releasing': 0,
     'Universal Pictures': 0,
     'Universal Pictures International': 0,
     'Walt Disney Studios Motion Pictures': 0,
     'Warner Bros.': 1
    })


y_testing = model.predict(data)
print(scaler_testy.inverse_transform(y_testing))
print("Predictions after denromalization:", scaler_testy.inverse_transform(y_pred))
```

```
Mean Squared Error: 0.02597806178602271
[[2.6540054e+11]]
Predictions after denromalization: [[ 36951238.80458038]
 [ 21348819.31522573]
 [-11139686.85258514]
 [ 42754370.34483511]
 [ 30502041.45201363]
 [ 18761151.88676185]
 [ 47115933.38107492]
 [ 46340871.62911883]
 [ 14517433.90398519]
 [ 39602727.85942113]
 [ 17884240.20256313]
 [ 41015049.67621395]
 [ 25314382.21011815]
 [ 15668459.92460897]
 [ 22160919.2257145 ]
 [ 26413043.17879603]
 [ 33768303.77357139]
 [ 33755429.97989671]
 [ 27494401.37946711]
 [ 33907857.35448853]
 [ 37019182.02547552]
 [ 39056634.32840344]
 [ 45222268.87668163]
 [ 32703790.67213768]
 [ 25221595.01015545]
 [ 35122420.83656026]
 [ 35187267.37293343]
 [ 29277681.1297293 ]
 [ 37713185.60696513]
 [ 25322042.41096588]
 [ 38467913.47289063]
 [ 43579680.24043211]
 [ 24612068.35127961]
 [ 27723846.44153863]
 [ 23644583.898509  ]
 [ 61982001.84462436]
 [ 28216264.71281756]
 [ 25914212.67156998]
 [ 43695278.58739015]
 [ 58123632.93042068]
 [ 56453834.61881153]
 [ 41288737.80512356]
 [ 36766083.33794712]
 [ 23425281.22205063]
 [ 35971713.26737578]
 [ 42797287.58633655]
 [ 15885498.62578608]
 [ 43887701.28820468]
 [ 26432984.96255535]
 [ 36684633.72262842]
 [ 30752939.83221895]
 [ 35754674.56619867]
 [ 22655674.21881815]
 [ 11313426.3666253 ]
 [ 54800247.94129567]
 [ 48926181.65080009]
 [ 29715395.25024821]
 [ 60912033.99446703]
```

[34870694.8919153]
[31734189.31106207]
[45819002.6536115]
[45601423.91879166]
[14689692.63053127]
[39512676.99132062]
[35405678.81699395]
[22769155.65747119]
[41717459.22800843]
[37252432.36074591]
[39126694.45760356]
[30599374.92453159]
[31971012.69653897]
[31971012.69653897]
[25062656.26547319]
[22181806.68044663]
[38583159.09927934]
[28339888.53708375]
[14121203.9553151]
[48188997.9157542]
[20526736.55976984]
[35344170.12103941]
[27797571.28382654]
[49363895.77131901]
[52044270.14543591]
[52708947.1432609]
[38208527.32739795]
[30972242.50867731]
[14724454.99753197]
[43176506.29640479]
[38409560.63667522]
[38680574.3667798]
[26103538.68786255]
[28216264.71281756]
[30328424.06028945]
[29837428.12214683]
[29989228.84209853]
[30329673.15805767]
[30679977.98218761]
[45212344.70055263]
[45176443.03183635]
[39074962.87080172]
[27316073.40444089]
[44417974.62998129]
[41614734.67472507]
[56405199.71653165]
[30328424.06028945]
[25276745.34910462]
[39067743.93434246]
[25914212.67156998]
[44111326.32509216]
[27816869.8735973]
[34052452.98640294]
[797177.86823258]
[28619155.39215271]
[45802791.01951821]
[50237951.26947338]
[31237916.1121343]
[56729432.39839751]
[32490922.52061988]

[22125650.56893439]
[31677002.97553663]
[18046356.54349605]
[40186346.68677966]
[20040387.53697106]
[42855370.29906088]
[29212834.59335614]
[27011390.19708693]
[26964872.20311205]
[36540493.20352445]
[31870169.84177272]
[27347123.92974404]
[33004592.78345097]
[36723550.69097278]
[44942940.94742963]
[36869547.16896985]
[23209678.8354247]
[44784554.75616272]
[46697527.57917127]
[34160720.83222903]
[49190389.07987232]
[44001182.72685772]
[51701708.92117178]
[30502041.45201363]
[27559247.91584028]
[30328424.06028945]
[25077495.15668306]
[27281885.94851865]
[29370099.79087593]
[39899639.30125466]
[20924637.63431337]
[26763505.1023855]
[29034506.61832992]
[32031894.73785341]
[9708866.03624155]
[27967876.60859121]
[35297652.12706452]
[27233251.04623876]
[44657670.74085098]
[26894570.91801528]
[52708947.1432609]
[24915378.81575013]
[38167111.36157732]
[31134358.84961025]
[29541888.55880152]
[39528888.62541392]
[29585702.17750188]
[32388550.68790585]
[33355352.72039134]
[37484105.82135404]
[17491823.14390549]
[35669794.47490875]
[31971012.69653897]
[28535703.07876223]
[43316956.2545208]
[42797287.58633655]
[40937183.88322539]
[29406001.45959222]
[32463712.84526024]
[38905627.59340954]

```
 [ 24336470.57169364]
 [ 36733268.62490831]
 [ 57280627.95756947]
 [ 29454244.91701987]
 [ 21690636.37406831]
 [ 34319499.33274335]
 [ 12501977.68358994]
 [ 51941545.59215255]
 [ 32766520.30020586]
 [ 35838541.79409973]
 [ 26953711.46941604]
 [ 30328424.06028945]
 [ 40258412.1596121 ]
 [ 38021206.65473769]
 [ 27240469.98269803]
 [ 43771027.47207373]
 [ 25510478.629239  ]
 [ 46645795.99236943]
 [ 43341719.32185967]
 [ 43787093.6432263 ]
 [ 17911561.44259544]
 [ 47491897.64974261]]
```

```
C:\Users\malak\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarning: X has fe
ature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

In [ ]:

In [75]:
```
from sklearn.metrics import accuracy_score
model.score(X_test,y_test)
```

Out[75]:
```
0.1391197163978004
```

In [76]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Creating a Random Forest regression model
model = RandomForestRegressor(random_state=42)

# Training the model
model.fit(X_train, y_train.ravel())  # or y_train.squeeze()

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test.ravel(), y_pred)  # or y_test.squeeze()
print("Mean Squared Error:", mse)

#print("Prediction before denormalization:", y_pred)
print("Predictions after denormalization:", scaler_testy.inverse_transform([y_pred]))
```

```
Mean Squared Error: 0.0254232303217064
Predictions after denormalization: [[2.46461069e+07 2.25595157e+07 2.33345911e+07 3.1
7238967e+07
  2.52411749e+07 2.39686668e+07 2.98548727e+07 4.90916115e+07
  2.86400201e+07 4.19044511e+07 2.23140361e+07 3.55958666e+07
  2.70450482e+07 2.45047320e+07 2.32399897e+07 2.23277521e+07
  2.65858986e+07 2.78450276e+07 2.51174567e+07 2.68703211e+07
  4.59009726e+07 3.37895782e+07 3.17771900e+07 2.51745375e+07
  2.90258143e+07 4.36486115e+07 2.94651088e+07 3.05369421e+07
  2.49674183e+07 2.51817733e+07 4.06183320e+07 4.77117637e+07
  2.80649784e+07 2.84712714e+07 2.24545424e+07 8.89926080e+07
  2.72635451e+07 3.11280909e+07 4.51455345e+07 7.72373386e+07
  1.09064790e+08 2.79346699e+07 3.07768872e+07 2.42734626e+07
  2.91660401e+07 2.69112175e+07 2.26834681e+07 5.24964939e+07
  2.45757078e+07 4.47744328e+07 2.80936339e+07 3.39725625e+07
  2.28020721e+07 2.31075170e+07 8.41909552e+07 4.96787259e+07
  2.58263297e+07 7.42772901e+07 5.09267279e+07 2.80009189e+07
  2.74675544e+07 4.11154110e+07 2.24875549e+07 2.43649614e+07
  4.17246125e+07 2.25508058e+07 3.04557801e+07 3.12959550e+07
  3.65586150e+07 2.81196454e+07 2.62821333e+07 2.62821333e+07
  2.64985765e+07 2.34740821e+07 5.22566662e+07 3.61697076e+07
  2.32191628e+07 6.00166141e+07 2.32550570e+07 3.83635014e+07
  2.62044709e+07 5.63551968e+07 4.48759546e+07 3.95434873e+07
  3.73941168e+07 2.72603913e+07 2.41082834e+07 2.63659290e+07
  3.12953686e+07 2.94032294e+07 2.40675147e+07 2.72635451e+07
  2.72575169e+07 2.24491445e+07 3.13952771e+07 2.87314030e+07
  2.79729843e+07 4.67549207e+07 3.48128138e+07 2.69947206e+07
  2.47763318e+07 5.80973797e+07 3.40157015e+07 9.87048516e+07
  2.72575169e+07 2.58789347e+07 3.21821806e+07 3.11280909e+07
  3.02840985e+07 3.21210578e+07 2.67383054e+07 2.25685734e+07
  2.59695683e+07 2.82041306e+07 7.48998834e+07 2.74334988e+07
  9.00157840e+07 2.90451731e+07 2.57753993e+07 2.85662069e+07
  2.22530058e+07 3.21067011e+07 2.48239216e+07 4.48244907e+07
  3.04555742e+07 2.49711243e+07 2.25539107e+07 2.77683375e+07
  3.39292829e+07 2.60824965e+07 2.85510390e+07 2.92944332e+07
  3.73825835e+07 2.57604340e+07 2.42171324e+07 3.84265016e+07
  4.36665586e+07 2.48811866e+07 6.89290918e+07 5.36587900e+07
  7.26015987e+07 2.52411749e+07 2.52281958e+07 2.72575169e+07
  2.71136798e+07 3.61147355e+07 2.65944994e+07 2.90879876e+07
  2.73300893e+07 2.67685779e+07 3.07146524e+07 3.16409085e+07
  2.65125394e+07 2.66699782e+07 2.61557911e+07 3.65039572e+07
  3.82663975e+07 2.33234178e+07 3.95434873e+07 2.62961294e+07
  2.43615622e+07 2.81246901e+07 2.29513973e+07 2.43894473e+07
  2.55656766e+07 2.81926466e+07 2.90550908e+07 3.08324232e+07
  2.40687748e+07 3.33042667e+07 2.62821333e+07 2.38793803e+07
  3.79715687e+07 2.69112175e+07 3.14944355e+07 2.75004707e+07
  2.54379988e+07 3.16605584e+07 2.56277999e+07 3.17015898e+07
  8.58383714e+07 2.60676940e+07 2.61309956e+07 2.26138674e+07
  2.35918281e+07 7.13158789e+07 3.11610539e+07 2.98456786e+07
  2.47953466e+07 2.72575169e+07 2.92435565e+07 2.40274615e+07
  2.29316885e+07 5.03336947e+07 2.85720899e+07 2.95630682e+07
  3.54481950e+07 2.45634811e+07 2.19944279e+07 6.86860706e+07]]
```

In [77]:
```
from sklearn.metrics import accuracy_score
model.score(X_test,y_test)
```

Out[77]:
```
0.15750613307069639
```

In [2]:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox


# Load the dataset
df = pd.read_csv("Top_Highest_Openings.csv")

df = df.fillna(df.Theaters.mean())

df.isnull().sum()

# Handle Duplicates
df = df.drop_duplicates()

# Convert Distributor column
dummies = pd.get_dummies(df['Distributor'])
df = pd.concat([df, dummies], axis=1)
df.drop(['Release','Date','Total Gross','% of Total','Average','Distributor'],axis='co

# Normalize the dataset
scaler_trainx = MinMaxScaler()
scaler_trainy = MinMaxScaler()
scaler_testx = MinMaxScaler()
scaler_testy = MinMaxScaler()

target = df['Opening']
df = df.drop('Opening', axis=1)

X_train, X_test, y_train, y_test = train_test_split(df, target, test_size=0.2, random_

X_train = scaler_trainx.fit_transform(pd.DataFrame(X_train))
y_train = scaler_trainy.fit_transform(pd.DataFrame(y_train))
X_test = scaler_testx.fit_transform(pd.DataFrame(X_test))
y_test = scaler_testy.fit_transform(pd.DataFrame(y_test))


def predict_revenue():
    theaters = int(theaters_entry.get())
    selected_distributor = distributor_var.get()

    data = pd.DataFrame({
        'Theaters': [theaters],
        '-': [0],
        '20th Century Studios': [0],
        'Columbia Pictures': [0],
        'Dimension Films': [0],
        'DreamWorks': [0],
        'DreamWorks Distribution': [0],
        'FilmDistrict': [0],
        'Focus Features': [0],
```

```
        'Lionsgate Films': [0],
        'Metro-Goldwyn-Mayer': [0],
        'Miramax': [0],
        'New Line Cinema': [0],
        'Newmarket Films': [0],
        'Paramount Pictures': [0],
        'Relativity Media': [0],
        'Revolution Studios': [0],
        'STX Entertainment': [0],
        'Screen Gems': [0],
        'Sony Pictures Releasing': [0],
        'Summit Entertainment': [0],
        'The Weinstein Company': [0],
        'TriStar Pictures': [0],
        'Twentieth Century Fox': [0],
        'United Artists Releasing': [0],
        'Universal Pictures': [0],
        'Universal Pictures International': [0],
        'Walt Disney Studios Motion Pictures': [0],
        'Warner Bros.': [0]
    })
    data[selected_distributor] = 1

    data_scaled = scaler_testx.transform(data)

    if model_var.get() == "Linear Regression":
        y_testing_scaled = model_lr.predict(data_scaled)
    elif model_var.get() == "Decision Tree Regression":
        y_testing_scaled = model_dt.predict(data_scaled)
    else:  # Random Forest Regression
        y_testing_scaled = model_rf.predict(data_scaled)

    # Reshape y_testing_scaled to 2D array
    y_testing_scaled = y_testing_scaled.reshape(-1, 1)

    y_testing = scaler_testy.inverse_transform(y_testing_scaled)
    average_predicted_revenue = y_testing.mean()

    result_label.config(text=f"Predicted Revenue: ${average_predicted_revenue:,.2f}")


def change_model(event):
    if model_var.get() == "Linear Regression":
        model_lr.fit(X_train, y_train)
    elif model_var.get() == "Decision Tree Regression":
        model_dt.fit(X_train, y_train)
    else:  # Random Forest Regression
        model_rf.fit(X_train, y_train.ravel())

# Initialize models
model_lr = LinearRegression()
model_dt = DecisionTreeRegressor(max_depth=100, min_samples_split=6, random_state=42)
model_rf = RandomForestRegressor(random_state=42)

root = tk.Tk()
root.title("Movie Revenue Prediction")
root.geometry("800x600")
```

```
def exit_program():
    root.destroy()




bg_image = Image.open("machine5.jpg")
bg_image = bg_image.resize((1500, 800), Image.LANCZOS)
bg_image_tk = ImageTk.PhotoImage(bg_image)
canvas = tk.Canvas(root, width=1500, height=800)
canvas.grid(row=0, column=0, columnspan=4, sticky="nsew")
canvas.create_image(0, 0, image=bg_image_tk, anchor="nw")
canvas.bg_image_tk = bg_image_tk


model_label = tk.Label(root, text="Select Model:")
model_label_canvas=canvas.create_window( 400, 10,  anchor = "nw", window = model_label

model_var = tk.StringVar()
model_var.set("Linear Regression")
model_dropdown = ttk.Combobox(root, textvariable=model_var, values=["Linear Regressior
model_dropdown_canvas=canvas.create_window( 600, 10,  anchor = "nw", window = model_dr
model_dropdown.bind("<<ComboboxSelected>>", change_model)

theaters_label = tk.Label(root, text="Number of Theaters:")
theaters_label_canvas=canvas.create_window( 400, 60,  anchor = "nw", window = theaters

theaters_entry = ttk.Entry(root)
theaters_entry_canvas=canvas.create_window( 600, 60,  anchor = "nw", window = theaters

distributor_label = tk.Label(root, text="Select Distributor:")
distributor_label_canvas=canvas.create_window( 400, 110,  anchor = "nw", window = dist

distributor_var = tk.StringVar()

distributor_dropdown = ttk.Combobox(root, textvariable=distributor_var, values=[
    '20th Century Studios', 'Columbia Pictures', 'Dimension Films', 'DreamWorks',
    'DreamWorks Distribution', 'FilmDistrict', 'Focus Features', 'Lionsgate Films',
    'Metro-Goldwyn-Mayer', 'Miramax', 'New Line Cinema', 'Newmarket Films',
    'Paramount Pictures', 'Relativity Media', 'Revolution Studios', 'STX Entertainment
    'Screen Gems', 'Sony Pictures Releasing', 'Summit Entertainment', 'The Weinstein C
    'TriStar Pictures', 'Twentieth Century Fox', 'United Artists Releasing',
    'Universal Pictures', 'Universal Pictures International', 'Walt Disney Studios Mot
    'Warner Bros.'
], state='readonly', width=50)  # Set the width and state
distributor_dropdown_canvas=canvas.create_window( 600, 110,  anchor = "nw", window =di




predict_button = tk.Button(root, text="Predict Revenue", command=predict_revenue , bg=
predict_button_canvas=canvas.create_window( 500, 160,  anchor = "nw", window =predict_

result_label = tk.Label(root, text="")
result_label_canvas=canvas.create_window( 500, 210,  anchor = "nw", window =result_lab

exit_button = tk.Button(root, text="Exit", command=exit_program, bg='darkorchid3', act
buttonexit_canvas = canvas.create_window( 1200, 600, anchor = "nw", window =exit_butto
```

```
root.mainloop()
```

In [2]:
```
pip install pyinstaller
```

```
Requirement already satisfied: pyinstaller in c:\users\malak\anaconda3\lib\site-packa
ges (6.6.0)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: setuptools>=42.0.0 in c:\users\malak\anaconda3\lib\sit
e-packages (from pyinstaller) (68.0.0)
Requirement already satisfied: altgraph in c:\users\malak\anaconda3\lib\site-packages
(from pyinstaller) (0.17.4)
Requirement already satisfied: pyinstaller-hooks-contrib>=2024.3 in c:\users\malak\an
aconda3\lib\site-packages (from pyinstaller) (2024.6)
Requirement already satisfied: packaging>=22.0 in c:\users\malak\anaconda3\lib\site-p
ackages (from pyinstaller) (23.1)
Requirement already satisfied: pefile>=2022.5.30 in c:\users\malak\anaconda3\lib\site
-packages (from pyinstaller) (2023.2.7)
Requirement already satisfied: pywin32-ctypes>=0.2.1 in c:\users\malak\anaconda3\lib
\site-packages (from pyinstaller) (0.2.2)
```

In [ ]:
```

```