

گزارش جامع تحلیل مدل‌های پیش‌بینی درآمد گیشه سینما

مقدمه

پیش‌بینی درآمد گیشه فیلم‌ها یکی از مهم‌ترین چالش‌های صنعت سینما است. مدل‌های یادگیری ماشین می‌توانند با استفاده از ویژگی‌های مختلف فیلم، درآمد گیشه را تخمین بزنند. در این گزارش، عملکرد سه مدل مختلف رگرسیون خطی، درخت تصمیم و جنگل تصادفی برای پیش‌بینی میزان فروش فیلم‌ها مقایسه شده است. هدف این بررسی انتخاب مدلی با دقت بالا و قابلیت تعمیم مناسب است.

مقایسه عملکرد مدل‌ها

مدل	ضریب تعیین (R^2)	میانگین مربعات خطا (MSE)
رگرسیون خطی	0.5597	2.22e+16
درخت تصمیم	0.4583	2.73e+16
جنگل تصادفی	0.6483	1.77e+16
بهترین جنگل تصادفی (GridSearch)	0.6483	1.77e+16

♦ نتیجه‌گیری: مدل جنگل تصادفی بهینه‌شده بهترین عملکرد را نشان داده است. این مدل کمترین میزان خطا (MSE) و بالاترین ضریب تعیین (R^2) را دارد که نشان‌دهنده دقت بالاتر و تعمیم‌پذیری بهتر آن نسبت به سایر مدل‌ها است. مدل درخت تصمیم ضعیف‌ترین عملکرد را داشت و رگرسیون خطی عملکرد متوسطی ارائه داد.

تحلیل نمودارها

۱. نمودار مقایسه پیش‌بینی با مقدار واقعی

این نمودار توزیع مقادیر واقعی و پیش‌بینی‌شده را برای مدل جنگل تصادفی بهینه‌شده نشان می‌دهد:

- نقاط آبی: مقادیر پیش‌بینی‌شده توسط مدل.
- خط قرمز: خط ایده‌آل که نشان می‌دهد پیش‌بینی‌ها باید چقدر به مقادیر واقعی نزدیک باشند.
- مشاهده می‌شود که بسیاری از نقاط نزدیک به خط قرمز قرار دارند که نشان‌دهنده دقت بالای مدل است.
- در فیلم‌هایی که درآمد گیشه بسیار بالایی دارند، مدل گاهی مقدار را کمتر از واقعیت پیش‌بینی کرده است که نشان‌دهنده سختی پیش‌بینی فیلم‌های پرفروش است.

۲. نمودار اهمیت ویژگی‌ها

این نمودار نشان می‌دهد که کدام ویژگی‌ها بیشترین تأثیر را در پیش‌بینی درآمد گیشه داشته‌اند:

- **بودجه تولید (rt_production_budget):** مهم‌ترین عامل تأثیرگذار که نشان می‌دهد هر چه فیلمی بودجه بالاتری داشته باشد، احتمال فروش بالاتر آن بیشتر است.
- **امتیاز مخاطبان (rt_audience_score):** دومین عامل تأثیرگذار، که نشان می‌دهد نظرات و بازخوردهای مخاطبان تأثیر مستقیمی بر فروش فیلم دارد.
- **مدت زمان فیلم (rt_runtime):** مدت زمان طولانی‌تر می‌تواند بر تجربه مخاطب تأثیر بگذارد، اما تأثیر آن کمتر از بودجه و امتیاز مخاطبان است.
- **سال انتشار (release_year):** سال‌های جدیدتر به‌طور متوسط درآمد بیشتری دارند که ممکن است به دلیل تغییرات در بازار و افزایش فروش جهانی باشد.
- **عوامل دیگر مانند حضور بازیگر مشهور و کارگردان شناخته‌شده تأثیر کمی دارند،** که نشان می‌دهد موفقیت فیلم بیش از آنکه به شهرت عوامل بستگی داشته باشد، به کیفیت کلی فیلم و بودجه وابسته است.

نقاط قوت و ضعف مدل‌ها

مزایا:

- مدل جنگل تصادفی دارای دقت بالاتر و تعمیم‌پذیری مناسب است.
- این مدل می‌تواند الگوهای پیچیده بین متغیرها را یاد بگیرد و بهتر از مدل‌های خطی عمل می‌کند.

معایب:

- مدل جنگل تصادفی برای فیلم‌هایی که درآمد بسیار بالایی دارند، گاهی مقدار کمتری را پیش‌بینی می‌کند.
- پیچیدگی محاسباتی این مدل بیشتر است و زمان پردازش طولانی‌تری دارد.

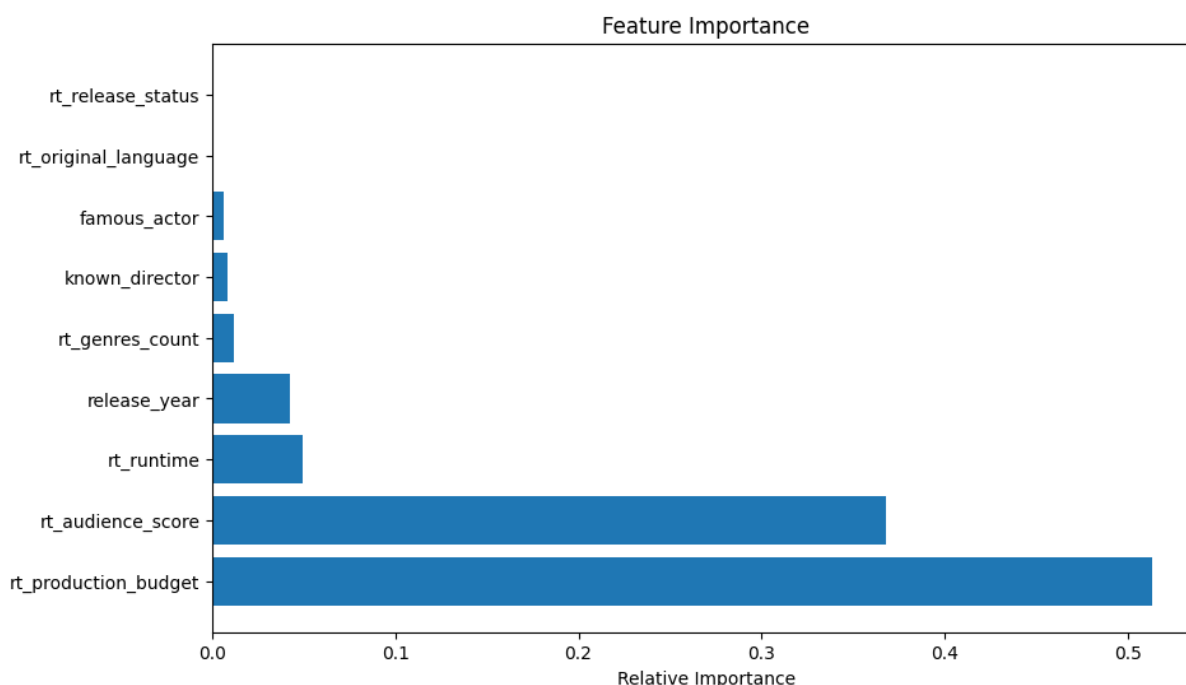
پیشنهادهای برای بهبود عملکرد مدل

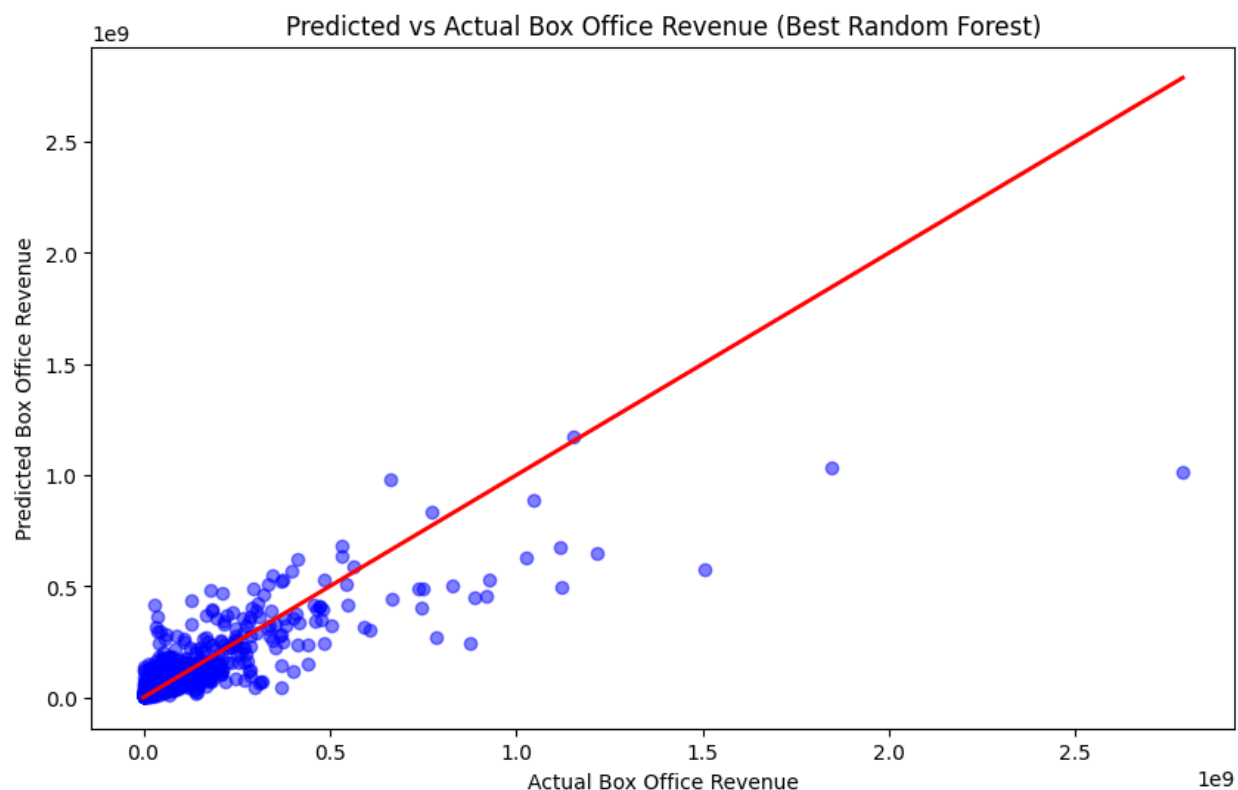
1. **افزایش حجم داده‌ها:** مدل می‌تواند با داشتن داده‌های بیشتر دقت بالاتری داشته باشد. به‌ویژه برای فیلم‌هایی که درآمد بسیار بالایی دارند، اضافه کردن داده‌های مشابه می‌تواند مدل را بهبود بخشد.
2. **افزودن ویژگی‌های جدید:** برخی عوامل مانند میزان تبلیغات، حضور در جشنواره‌ها، نوع اکران (سینما یا استریمینگ) و تحلیل نقدهای منتقدان می‌توانند نقش مهمی در تعیین میزان فروش فیلم داشته باشند.

3. استفاده از مدل‌های پیشرفته‌تر: روش‌هایی مانند **XGBoost**، **LightGBM** یا شبکه‌های عصبی عمیق می‌توانند الگوهای پیچیده‌تری را شناسایی کرده و پیش‌بینی دقیق‌تری ارائه دهند.
4. بررسی تعامل بین متغیرها: تحلیل ارتباط بین ویژگی‌ها می‌تواند در بهبود مدل مؤثر باشد. به عنوان مثال، تعامل بین بودجه تولید و ژانر فیلم می‌تواند به درک بهتری از موفقیت فیلم کمک کند.
5. تنظیمات بهتر مدل: می‌توان از روش‌های بهینه‌سازی پیشرفته‌تر مانند **Bayesian Optimization** برای یافتن بهترین ترکیب از پارامترهای مدل استفاده کرد.

نتیجه‌گیری نهایی

♦ جنگل تصادفی بهینه‌شده بهترین مدل برای پیش‌بینی درآمد گیشه فیلم‌ها بود. ♦ بودجه تولید و امتیاز مخاطبان دو عامل کلیدی در تعیین میزان فروش فیلم‌ها بودند. ♦ برای بهبود مدل می‌توان داده‌های بیشتری جمع‌آوری کرد، ویژگی‌های جدید اضافه نمود و از مدل‌های پیشرفته‌تری بهره گرفت.





توضیحات کد

بارگذاری و پردازش داده‌ها

```
python
CopyEdit
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
```

2. بارگذاری داده‌ها

```
python
CopyEdit
df_movies = pd.read_csv(r"rotten_tomatoes_5000_movies.csv")
```

```
df_credit = pd.read_csv(r"rotten_tomatoes_5000_credits.csv")
```

3. پیش پردازش داده‌ها

3.1. حذف مقادیر گمشده

```
python
CopyEdit
df_movies['rt_tagline'].fillna('', inplace=True)
df_movies['rt_website'].fillna('', inplace=True)
```

❗ مقادیر NaN در ستون‌های rt_tagline و rt_website را با مقدار خالی جایگزین می‌کند.

```
python
CopyEdit
df_movies['rt_box_office'] = df_movies['rt_box_office'].replace(0, np.nan)
df_movies['rt_production_budget'] =
df_movies['rt_production_budget'].replace(0, np.nan)
```

❗ مقادیر صفر را در ستون‌های rt_box_office و rt_production_budget با NaN جایگزین می‌کند، چون مقدار صفر در اینجا منطقی نیست.

3.2. استخراج ویژگی‌های جدید

3.2.1. حضور بازیگران مشهور

```
python
CopyEdit
def famous_actor_in_movie(row):
    famous_actors = ['Tom Hanks', 'Leonardo DiCaprio', 'Brad Pitt', 'Scarlett
Johansson', 'Robert Downey Jr.']
    for actor in famous_actors:
        if actor in row['rt_actors']:
            return 1
    return 0

df_credit['rt_actors'] = df_credit['rt_actors'].apply(lambda x: str(x))
df_movies['famous_actor'] = df_credit.apply(famous_actor_in_movie, axis=1)
```

❗ بررسی می‌کند که آیا فیلم شامل بازیگران مشهور است یا خیر. اگر بله، مقدار 1 و در غیر این صورت 0.

3.2.2. حضور کارگردانان معروف

```
python
CopyEdit
def known_director_in_movie(row):
    known_directors = ['Steven Spielberg', 'Martin Scorsese', 'Christopher
Nolan', 'Quentin Tarantino', 'Ridley Scott']
```

```

for director in known_directors:
    if director in row['rt_staff']:
        return 1
    return 0

df_credit['rt_staff'] = df_credit['rt_staff'].apply(lambda x: str(x))
df_movies['known_director'] = df_credit.apply(known_director_in_movie,
axis=1)

```

بررسی می‌کند که آیا فیلم توسط یکی از کارگردانان مشهور ساخته شده است یا خیر. 🚫

3.2.3. تعداد ژانرهای فیلم

```

python
CopyEdit
df_movies['rt_genres_count'] = df_movies['rt_genres'].apply(lambda x:
len(x.split(',')))

```

تعداد ژانرهای هر فیلم را محاسبه می‌کند. 🚫

3.2.4. تبدیل تاریخ انتشار به سال

```

python
CopyEdit
df_movies['rt_release_date'] = pd.to_datetime(df_movies['rt_release_date'],
errors='coerce')
df_movies['release_year'] = df_movies['rt_release_date'].dt.year

```

تاریخ انتشار را به فرمت datetime تبدیل کرده و سال انتشار را استخراج می‌کند. 🚫

3.2.5. تبدیل داده‌های دسته‌ای به عددی

```

python
CopyEdit
le = LabelEncoder()
df_movies['rt_original_language'] =
le.fit_transform(df_movies['rt_original_language'])
df_movies['rt_release_status'] =
le.fit_transform(df_movies['rt_release_status'])

```

LabelEncoder داده‌های متنی (مثل زبان اصلی و وضعیت انتشار) را به مقدار عددی تبدیل می‌کند. 🚫


4. آماده‌سازی داده‌ها برای مدل‌سازی

```

python
CopyEdit


```

```
features = ['rt_production_budget', 'rt_audience_score', 'rt_runtime',  
            'famous_actor', 'known_director', 'rt_genres_count', 'release_year',  
            'rt_original_language', 'rt_release_status']  
X = df_movies[features]  
y = df_movies['rt_box_office']
```

انتخاب ویژگی‌های مهم و تعیین متغیر هدف. (rt_box_office) 


4.1. جایگذاری مقادیر گم‌شده

```
python  
CopyEdit  
from sklearn.impute import SimpleImputer  
imputer = SimpleImputer(strategy='mean')  
X_imputed = imputer.fit_transform(X)
```

جایگذاری مقادیر NaN با مقدار میانگین هر ستون. 

4.2. تقسیم داده‌ها به مجموعه آموزش و آزمون


```
python  
CopyEdit  
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y,  
                                                    test_size=0.2, random_state=42)
```

۸۰٪ داده‌ها برای آموزش و ۲۰٪ برای آزمایش مدل‌ها استفاده می‌شوند. 

5. آموزش مدل‌های یادگیری ماشین

5.1. رگرسیون خطی

```
python  
CopyEdit  
lr = LinearRegression()  
lr.fit(X_train, y_train)  
y_pred_lr = lr.predict(X_test)  
mse_lr = mean_squared_error(y_test, y_pred_lr)  
r2_lr = r2_score(y_test, y_pred_lr)
```

مدل رگرسیون خطی آموزش داده شده و عملکرد آن ارزیابی می‌شود. 

5.2. درخت تصمیم

```
python  
CopyEdit  
dt = DecisionTreeRegressor(random_state=42)  
dt.fit(X_train, y_train)  
y_pred_dt = dt.predict(X_test)  
mse_dt = mean_squared_error(y_test, y_pred_dt)
```

```
r2_dt = r2_score(y_test, y_pred_dt)
```

🔪 درخت تصمیم برای پیش‌بینی درآمد فیلم استفاده می‌شود.

5.3. جنگل تصادفی

```
python
CopyEdit
rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
```

🔪 جنگل تصادفی به عنوان مدل پیچیده‌تر برای بهبود دقت استفاده می‌شود.

6. تنظیمات پیشرفته (Hyperparameter Tuning)

```
python
CopyEdit
param_grid = {'n_estimators': [50, 100, 200], 'max_depth': [10, 20, None],
              'min_samples_split': [2, 5, 10]}
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3,
                           n_jobs=-1, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_
```

🔪 جستجوی شبکه‌ای (GridSearchCV) برای یافتن بهترین تنظیمات RandomForest.

7. ارزیابی و تحلیل نتایج

```
python
CopyEdit
print(f"Linear Regression - MSE: {mse_lr}, R2: {r2_lr}")
print(f"Decision Tree - MSE: {mse_dt}, R2: {r2_dt}")
print(f"Random Forest - MSE: {mse_rf}, R2: {r2_rf}")
```

🔪 مقایسه عملکرد مدل‌ها بر اساس MSE و R^2 .