Here is a Linear Regression model on Steam by Valve games that will take game features described in the dataset to determine/predict the initial price of a game.

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
file_path = 'games-features-edit.csv'
games_df = pd.read_csv(file_path)

# Select features and target variable
features = games_df[['Metacritic', 'RecommendationCount', 'IsFree', 'GenreIsIndie',
                     'GenreIsAction', 'GenreIsAdventure', 'GenreIsCasual',
                     'GenreIsStrategy', 'GenreIsRPG', 'GenreIsSimulation',
                     'GenreIsEarlyAccess', 'GenreIsFreeToPlay', 'GenreIsSports',
                     'GenreIsRacing', 'GenreIsMassivelyMultiplayer']]
target = games_df['PriceInitial']

# Convert boolean features to integers
features['IsFree'] = features['IsFree'].astype(int)
features.update(features.select_dtypes(include=[bool]).astype(int))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2

# Create a Linear Regression model
model = LinearRegression()

# Train 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)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 228.3735011320217
R-squared: 0.07278953615342554
```

```
C:\Users\joelr\AppData\Local\Temp\ipykernel_122260\804225047.py:19: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  features['IsFree'] = features['IsFree'].astype(int)
```

Lets try Random Forest to predict this better

In [ ]:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2

# Create a Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Feature importance
importance = rf_model.feature_importances_
feature_importance = pd.Series(importance, index=features.columns).sort_values(asce
print(feature_importance)
```

```
Mean Squared Error: 227.1354571938567
R-squared: 0.0778160707052814
RecommendationCount            0.468791
Metacritic                     0.150628
GenreIsAdventure               0.062450
GenreIsIndie                   0.061849
IsFree                         0.059259
GenreIsAction                  0.049132
GenreIsStrategy                0.028841
GenreIsCasual                  0.023578
GenreIsRPG                     0.021762
GenreIsSimulation              0.019877
GenreIsEarlyAccess             0.015346
GenreIsSports                  0.013983
GenreIsFreeToPlay              0.009052
GenreIsRacing                  0.008403
GenreIsMassivelyMultiplayer    0.007051
dtype: float64
```

In [ ]: