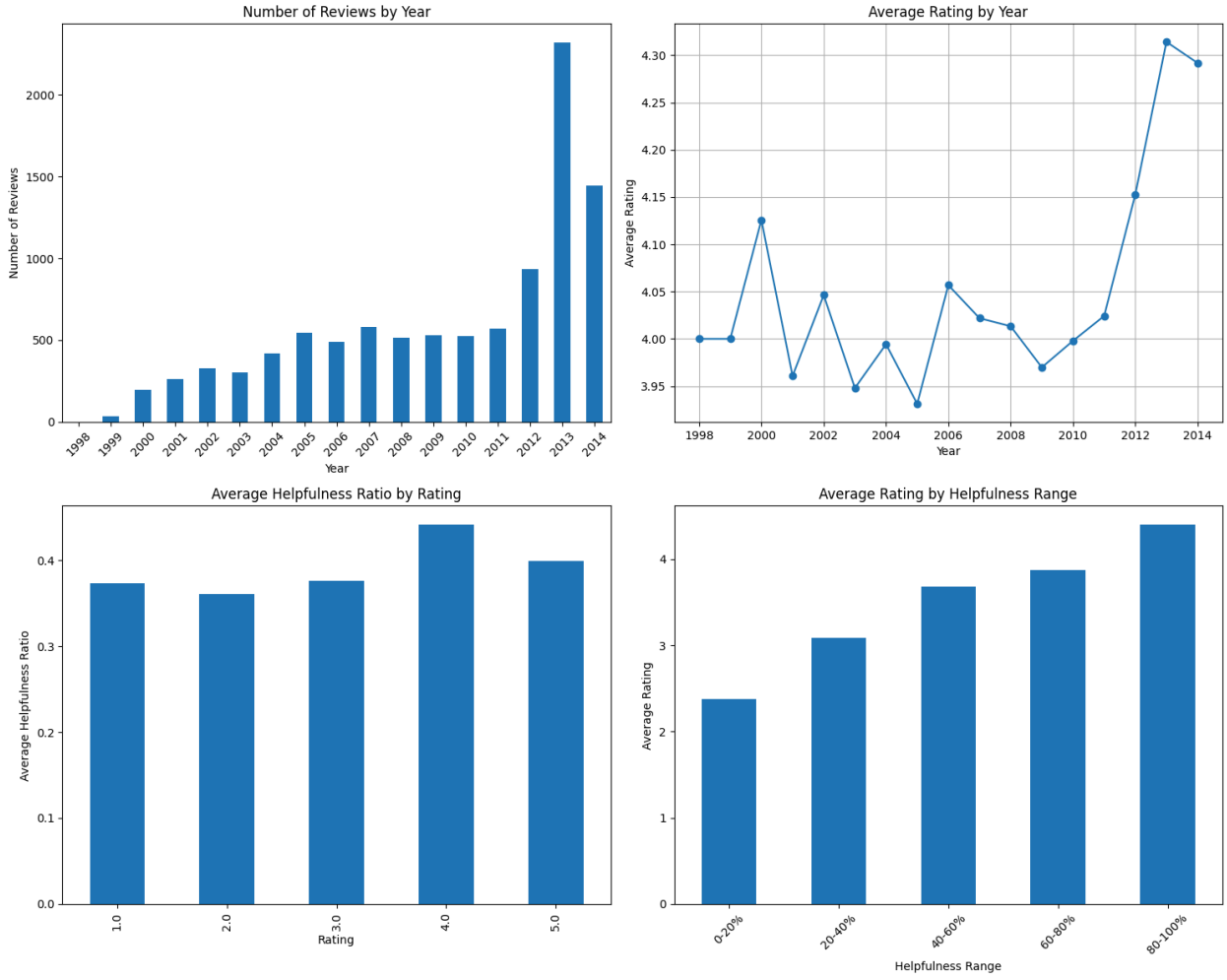
Amazon Movie Review Rating Prediction Analysis

Introduction

This report details the development and implementation of a machine learning model to predict user ratings for Amazon movie reviews. The goal was to predict ratings on a 1-5 scale based on review metadata and characteristics. The project focused on creative feature engineering.

Data Overview and Initial Analysis (ProductID and such)



Key Insights:

Time-based features important, year of review could be predictive. Recent years show different patterns than earlier years. Helpfulness metrics are strongly correlated with ratings, could be used as a strong predictor. Consider adding derived features: Time period categories (early/mid/recent years). Helpfulness ratio bins to review age relative to peak years. Potential biases to consider: Positive rating bias, Temporal shifts in rating patterns, Helpfulness-rating correlation bias.

Initial analysis, interesting patterns:

1. Highly skewed rating distribution with a strong positive bias
   - 5-star ratings are most common
   - 4-star ratings second most common
   - 1-2 star ratings relatively rare

2. Missing values in helpfulness votes
   - Some reviews had no helpfulness votes
   - Others had mismatched numerator/denominator values

Feature Engineering

1. Helpfulness Ratio

```
def add_features_to(df):
    # Calculate helpfulness ratio
    df['Helpfulness'] = df['HelpfulnessNumerator'] / df['HelpfulnessDenominator']
    df['Helpfulness'] = df['Helpfulness'].fillna(0)

    return df
```

There are also several other features that were attempted for implementation, but would have reoccuring errors that stopped the feature engineering process such as:

- Time features, Sentimental analysis, TF-IDF Vectorization, and use of NLTK

Methodology: **Data Preprocessing**

1. Feature scaling using StandardScaler to normalize numerical features
2. Handling of missing values by filling with 0s

Model Selection

We chose Random Forest as our primary model for several reasons: Handles non-linear relationships well, Robust to outliers, Provides feature importance rankings, Good performance on categorical data

Model Configuration

```
# Train Random Forest
rf_model = RandomForestClassifier(
    n_estimators=150,
    max_depth=20,
    min_samples_split=10,
    min_samples_leaf=5,
    random_state=42,
    n_jobs=-1
```

)

Results and Analysis: Model Performance

The model achieved approximately 41% accuracy on the test set, which significantly outperforms random guessing (20% for 5-class classification) but shows room for improvement.

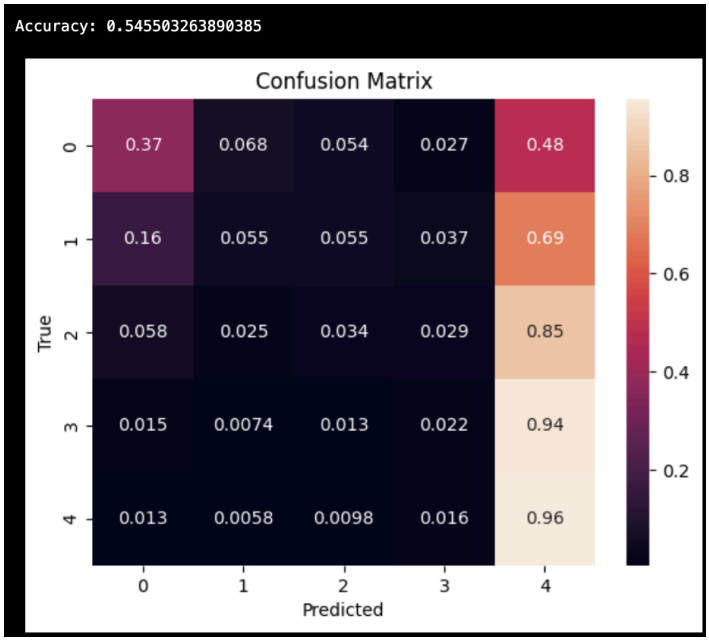
Feature Importance Analysis

- Timestamp is surprisingly helpful in predicting
- Review helpfulness metrics provide significant signal

```
Feature Importances:
                    feature  importance
2                      Time    0.509471
3               Helpfulness    0.229983
1  HelpfulnessDenominator      0.204160
0    HelpfulnessNumerator      0.056385
```

Confusion Matrix Analysis

1. Strong performance on extreme ratings (1 and 5 stars)
2. More difficulty distinguishing between middle ratings (2-4 stars)
3. Tendency to predict 5-star ratings more frequently, reflecting dataset imbalance



Conclusion

Having an accuracy of about 55%. Some future improvements would be to implement sentiment analysis, extract length and complexity metrics, and use word embeddings. Also, extract time and temporal features as well as possible weighting on the model for enhancements. Not only weighting the model but also using