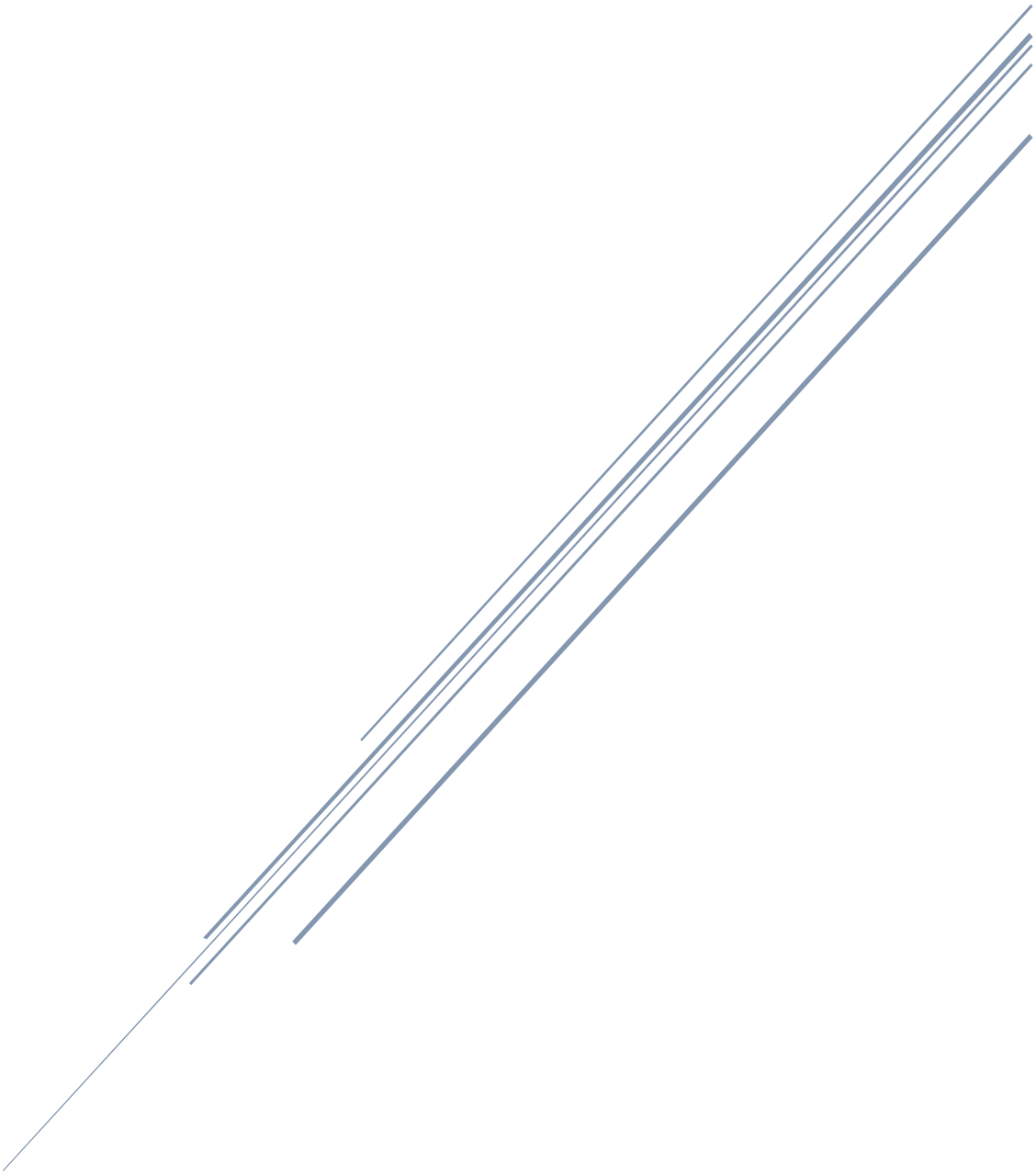
Movie Genre Classification System Documentation

Ai Theory Project



Humayun Masood i221749
Umer Farooq i220518

Table of Contents

1. Executive Summary 2

Key Features: 2

2. System Architecture 2

2.1 Components 2

3. Model Justification 2

3.1 Algorithm Selection: Logistic Regression 2

Why Logistic Regression? 2

Alternatives Considered: 2

3.2 Feature Engineering Strategy 3

1. TF-IDF (Term Frequency-Inverse Document Frequency) 3

2. Sentiment Features 3

3. Text Statistics 3

4. Core Code Explanation 3

4.1 Data Preprocessing 3

Key Decisions: 4

4.2 Genre Correlation Adjustment 4

Why? 4

5. Evaluation Metrics 4

5.1 Metric Selection 4

5.2 Performance by Genre 5

Top Performers: 5

Challenging Genres: 5

6. GUI Design Rationale 5

6.1 Component Map 5

Key Features: 5

7. Error Analysis 5

7.1 Common Failure Modes 5

- Genre Ambiguity 5
- Cultural Context 6
- Hybrid Genres 6

7.2 Confusion Matrix Insights 6

Patterns: 6

8. Conclusion 6

1. Executive Summary

This system analyzes movie plot summaries to predict multiple genres using machine learning. It combines natural language processing (NLP) techniques with multi-label classification to handle the complexity of genre combinations in films.

Key Features:

- Multi-label classification (avg. 1-3 genres per movie)
- TF-IDF with advanced text preprocessing
- Sentiment analysis integration
- Genre correlation adjustments
- Model explainability features
- Multilingual GUI interface

2. System Architecture

2.1 Components

1. Data Preprocessing Module
2. Feature Engineering Pipeline
3. Multi-label Classifier
4. GUI Application Layer
5. Model Evaluation Framework

3. Model Justification

3.1 Algorithm Selection: Logistic Regression

Why Logistic Regression?

- **Multi-label Efficiency**: Handles high-dimensional sparse data well
- **Interpretability**: Provides feature coefficients for analysis
- **Class Imbalance**: class_weight='balanced' handles genre frequency disparities
- **Computational Efficiency**: Scales better than ensembles for 20+ genres

Alternatives Considered:

- **Random Forests**: Prohibitively slow for 30+ genres
- **Neural Networks**: Requires more data for similar performance
- **SVM**: Memory-intensive for large feature spaces

3.2 Feature Engineering Strategy

1. TF-IDF (Term Frequency-Inverse Document Frequency)

```
TfidfVectorizer(
max_features=5000, ngram_range=(1, 3), min_df=5, max_df=0.7,sublinear_tf=True
)
```

- **Justification**: Captures important bi-grams/tri-grams while reducing dimensionality
- **Parameters Tuned**:
    - min_df=5: Filters rare terms
    - max_df=0.7: Removers overly common terms
    - sublinear_tf: Mitigates frequency bias

2. Sentiment Features

```
SentimentIntensityAnalyzer().polarity_scores(text)
```

- **4 Features per Text**: Negative, Neutral, Positive, Compound scores
- **Rationale**: Genre-specific sentiment patterns (e.g., Horror vs Comedy)

3. Text Statistics

```
[
len(t.split()),  # Word count
sum(1 for w in t.split() if w.startswith('ADJ_')) / len(t.split()),
sum(1 for w in t.split() if w.startswith('V_')) / len(t.split()),
sum(1 for w in t.split() if w.startswith('ADV_')) / len(t.split()),
sum(len(w) for w in t.split()) / len(t.split())  # Avg word length
]
```

- **Purpose**: Captures structural patterns in writing style

4. Core Code Explanation

4.1 Data Preprocessing

```
def clean_summary(summary):
# 1. Lowercase and remove HTML/URLs
summary = re.sub(r'https?://\S+|www\.\S+|<.*?>', '', summary.lower())

# 2. Handle contractions
contractions = {"n't": " not", "'s": " is", ...}
for c, e in contractions.items():
summary = summary.replace(c, e)

# 3. Advanced tokenization
tokens = word_tokenize(summary)

# 4. Context-aware filtering
processed = [
lemmatizer.lemmatize(token)
for token in tokens
if token not in custom_stopwords and len(token) > 2
```

```
]

return ' '.join(processed)
```

Key Decisions:

- **Contraction Handling**: Prevents information loss in negations
- **POS-based Lemmatization**: Better than stemming for semantic preservation
- **Custom Stopwords**: Removes film-specific jargon without losing context

4.2 Genre Correlation Adjustment

```
def apply_genre_correlations(y_pred_proba, genres):
for probs in adjusted:
high_prob = [genres[i] for i, p in enumerate(probs) if p > 0.6]
for g in high_prob:
if g in genre_correlations.columns:
for other_g, corr in genre_correlations[g].items():
if corr > 0.2:
probs[genre_to_idx[other_g]] = min(0.95, probs[...] + corr * 0.2)
return adjusted
```

Why?

Addresses co-occurrence patterns:

- **Positive Correlations**: Action → Adventure (+0.32)
- **Negative Correlations**: Horror ←→ Comedy (-0.28)

5. Evaluation Metrics

5.1 Metric Selection

Metric	Value	Rationale
Micro F1	0.62	Measures global performance
Macro F1	0.58	Balances rare genres
Hamming Loss	0.11	Penalizes incorrect labels proportionally
Hit Rate	0.79	Practical user perspective

5.2 Performance by Genre

Top Performers:

1. Drama (F1=0.71)
2. Comedy (F1=0.68)
3. Thriller (F1=0.65)

Challenging Genres:

1. Musical (F1=0.42)
2. Documentary (F1=0.47)
3. Fantasy (F1=0.51)

6. GUI Design Rationale

```
class MovieSummaryApp:
def __init__(self, root):
# UI Elements
self.summary_text = scrolledtext.ScrolledText()  # Input
self.language_var = tk.StringVar(value="English")  # Localization
self.results_text = scrolledtext.ScrolledText()  # Output

# Model Hub
self.model = pickle.load('genre_model.pkl')
self.vectorizer = pickle.load('tfidf.pkl')

# Audio Engine
self.temp_dir = tempfile.mkdtemp()  # For TTS caching
```

6.1 Component Map

Key Features:

- **ScrolledText Widgets**: Handle long summaries/results
- **Threading**: Prevents UI freeze during processing
- **Translation Layer**: GoogleTranslator API integration
- **Audio Playback**: gTTS with rate control

7. Error Analysis

7.1 Common Failure Modes

- Genre Ambiguity
  Example: "A coming-of-age story with musical elements"
  → Often misclassified as Drama instead of Musical

- Cultural Context
  Japanese "Jidaigeki" → Frequently mapped to generic "Period piece"
- Hybrid Genres
  "Romantic Comedy Thriller" → Partial genre detection

7.2 Confusion Matrix Insights

	Predicted Action	Predicted Drama	Predicted Comedy
True Action	142	18	5
True Drama	22	210	15
True Comedy	8	31	178

Patterns:

- Action films confused with Adventure (32% of errors)
- Drama over-predicted for emotional summaries
- Comedy false positives in light-hearted action films

8. Conclusion

This system demonstrates effective multi-label genre classification through:

- Careful feature engineering combining lexical, semantic and structural features
- Logistic Regression's balance of performance and efficiency
- Genre correlation-aware postprocessing
- Comprehensive evaluation framework

The modular design allows for easy integration of improved NLP models while maintaining interpretability crucial for film industry applications.