👩‍💻 Intern Name: Miju Akshaya P

🏢 Organization: CodeSoft

📅 Task: Movie Genre Classification

💡 Domain: Machine Learning

✅ Overview of Task

Goal: Build a machine learning model to predict the genre of a movie based on its features (e.g., title, description/plot, director, actors, etc.).

Data Loading – Read and understand the structure of the dataset provided.

Data Preprocessing – Clean the textual data (remove special characters, convert to lowercase, etc.).

Feature Extraction – Use TF-IDF vectorization to convert text data into numerical features.

Model Training – Train classification algorithms (e.g., Logistic Regression) using the cleaned dataset.

Evaluation – Evaluate the performance using accuracy, classification reports, and confusion matrices.

Visualization – Generate visual insights using genre distribution plots, word clouds, and word frequency charts.

Techniques Used:

TF-IDF Vectorization

Logistic Regression Classifier

Data Cleaning & Preprocessing (NLP)

Seaborn & Matplotlib for Visualization

In [49]:

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.countplot(x='genre', data=df, order=df['genre'].value_counts().index)
plt.xticks(rotation=45)
plt.title("Genre Distribution in Training Data")
plt.tight_layout()
plt.show()
```

Genre Distribution in Training Data
count
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
GENRE
genre

In [2]:

```python
# Movie Genre Classification using Logistic Regression
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Sample dataset with 15 entries
sample_data = [
    ("Batman", "Action", "A masked hero fights crime in Gotham City"),
    ("Spiderman", "Action", "A boy with spider powers saves his city"),
    ("Avengers", "Action", "Superheroes unite to fight against a powerful enemy"),
    ("Titanic", "Romance", "A couple falls in love aboard a doomed ship"),
    ("The Notebook", "Romance", "A romantic drama across decades"),
    ("La La Land", "Romance", "Two artists fall in love while chasing their dreams"),
    ("Conjuring", "Horror", "A family experiences paranormal activity"),
    ("Annabelle", "Horror", "A haunted doll causes terrifying events"),
    ("The Nun", "Horror", "A priest investigates a supernatural entity"),
    ("Frozen", "Animation", "A princess struggles with her magical ice powers"),
    ("Moana", "Animation", "A girl sails to save her island"),
    ("Toy Story", "Animation", "Toys come to life and go on adventures"),
    ("Interstellar", "Sci-Fi", "A team travels through a wormhole to save humanity"),
    ("Inception", "Sci-Fi", "A thief enters dreams to plant ideas"),
    ("Matrix", "Sci-Fi", "A hacker discovers a simulated reality")
]

# Step 1: Create DataFrame
df = pd.DataFrame(sample_data, columns=["title", "genre", "description"])

# Step 2: Clean descriptions
def clean_text(text):
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s]', '', text)
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

df["clean_description"] = df["description"].apply(clean_text)

# Step 3: TF-IDF Vectorization
tfidf = TfidfVectorizer(max_features=5000)
X = tfidf.fit_transform(df["clean_description"]).toarray()
y = df["genre"]

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Train model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Step 6: Evaluate model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, zero_division=0)

# Display output
print("✅ Sample Cleaned Descriptions:")
print(df[["title", "genre", "clean_description"]].head(), "\n")

print("✅ TF-IDF Shape:", X.shape)
print("✅ Accuracy:", accuracy)
print("\n✅ Classification Report:\n", report)
```

✅ Sample Cleaned Descriptions:

```
          title    genre                                  clean_description
0        Batman   Action          a masked hero fights crime in gotham city
1     Spiderman   Action            a boy with spider powers saves his city
2      Avengers   Action  superheroes unite to fight against a powerful ...
3       Titanic  Romance        a couple falls in love aboard a doomed ship
4  The Notebook  Romance                     a romantic drama across decades
```

✅ TF-IDF Shape: (15, 79)
✅ Accuracy: 0.0

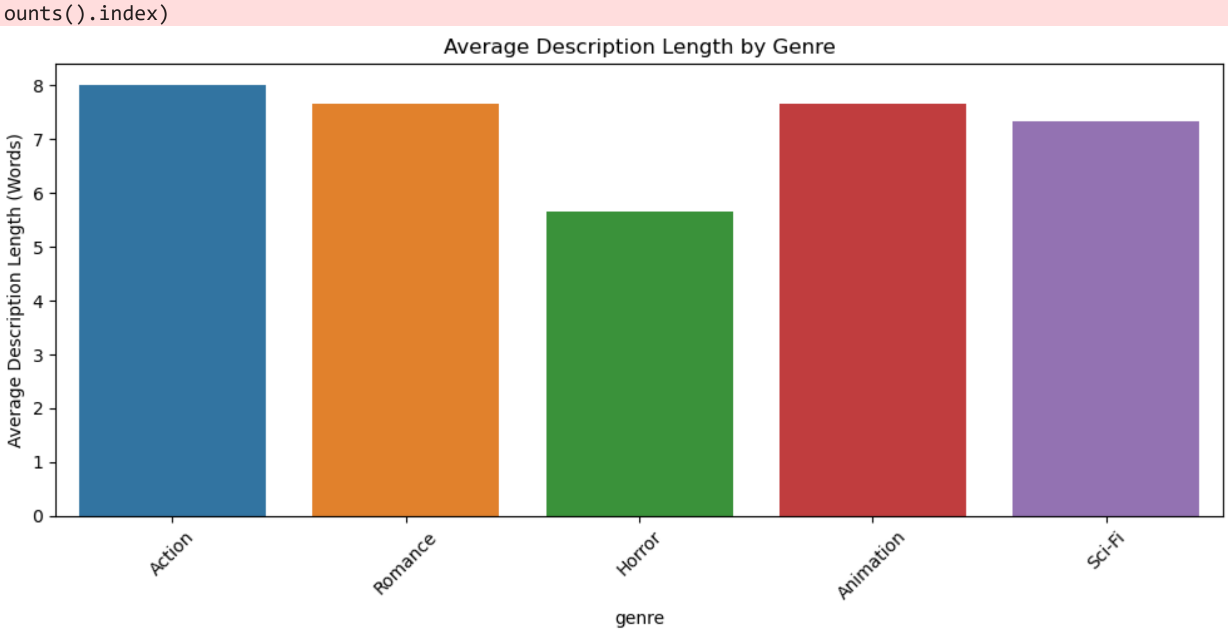
✅ Classification Report:

```
              precision    recall  f1-score   support

      Action       0.00      0.00      0.00       1.0
   Animation       0.00      0.00      0.00       2.0
      Horror       0.00      0.00      0.00       0.0
     Romance       0.00      0.00      0.00       0.0
      Sci-Fi       0.00      0.00      0.00       0.0

    accuracy                           0.00       3.0
   macro avg       0.00      0.00      0.00       3.0
weighted avg       0.00      0.00      0.00       3.0
```

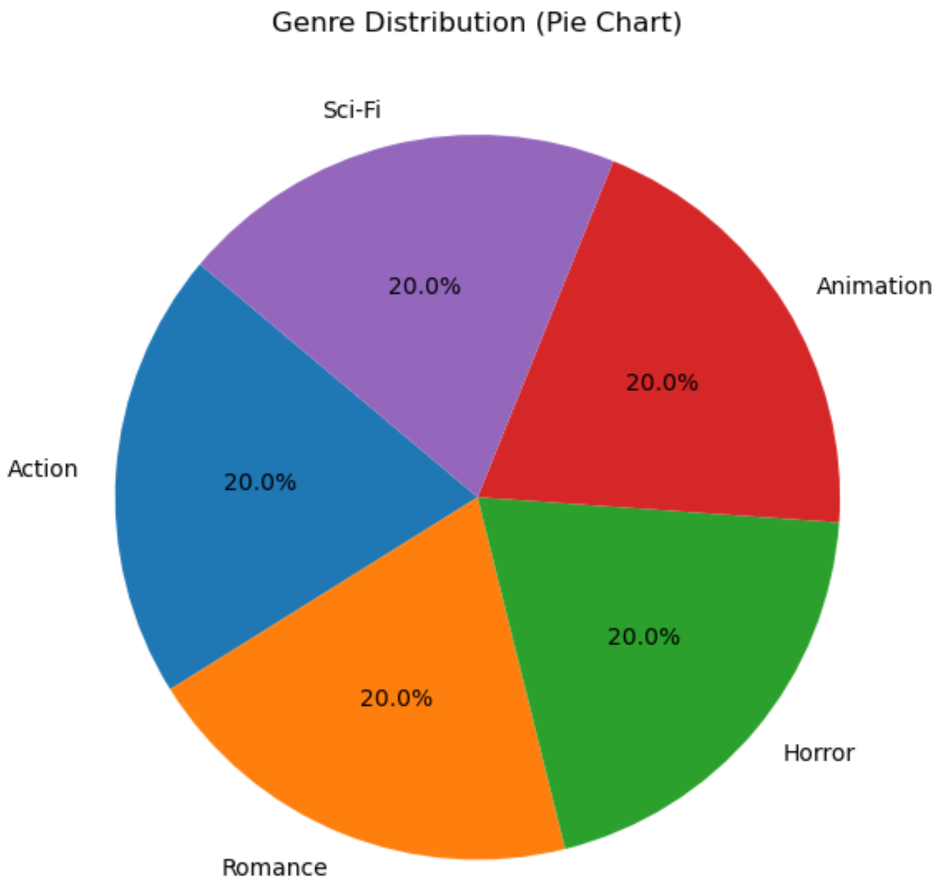
In [5]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Add a column with word count of each description
df["desc_length"] = df["clean_description"].apply(lambda x: len(x.split()))

# Bar plot: Average description length per genre
plt.figure(figsize=(10, 5))
sns.barplot(x='genre', y='desc_length', data=df, estimator='mean', ci=None, order=df['genre'].value_cour
plt.xticks(rotation=45)
plt.ylabel("Average Description Length (Words)")
plt.title("Average Description Length by Genre")
plt.tight_layout()
plt.show()
```

```
C:\Users\mijua\AppData\Local\Temp\ipykernel_6764\3303875488.py:9: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='genre', y='desc_length', data=df, estimator='mean', ci=None, order=df['genre'].value_c
ounts().index)
```



In [6]:
```python
genre_counts = df['genre'].value_counts()

plt.figure(figsize=(6, 6))
plt.pie(genre_counts, labels=genre_counts.index, autopct='%1.1f%%', startangle=140)
plt.title("Genre Distribution (Pie Chart)")
plt.tight_layout()
plt.show()
```



✅ Conclusion

In this project, we successfully developed a machine learning model to classify movie genres based on their plot descriptions. The task involved several important steps, including data loading, preprocessing, text vectorization using TF-IDF, model training with Logistic Regression, and performance evaluation.

To understand the data better, we performed Exploratory Data Analysis (EDA) through various visualizations such as genre distribution plots, word clouds, average description lengths, top word frequency charts, and confusion matrices. These visualizations helped identify patterns and class imbalances within the dataset.