Name=Saadullah Khan

Sentiment Analysis of IMDB Movie Reviews

IMDB dataset having 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing. So, predict the number of positive and negative reviews using either classification or deep learning algorithms. For more dataset information, please go through the following link, http://ai.stanford.edu/~amaas/data/sentiment/

Homework Assignment

- In the code we load the IMDB review data set. Perform stemming and remove the non-alphabets and stop words. Complete the rest of the following tasks.
- Create the TD matrix for both the training and testing data.
- Let n and p stand for the number of observations and the number of features, respectively. Print n and p.
- Fit a ridge logistic regression model (with the regularization given below) to the training TD matrix.
- Do the above ridge logistic regressions for $C = 0.001, C = 0.1, C = 10$.
- Use the TD-logistic-model to make predictions, and to compute the true and false positive rate for the testing data for thresholds $\theta \in [0.01, 0.02, 0.03, \cdots, 0.96, 0.97, 0.98, 0.99]$.
- Use the last two parts, to compute the train and test AUC, and to plot the ROC for the three models on the same plot, and to report the AUCs on the plot. In other words, the train figure whould be the ROCs for the three models with three Cs, leading to three ROC's on the same graph with appropiate labeling. Do the same thing for the test data. The plot should be a 1 x 2 panel.
- Also, make a 2×3 table that reports the train and test error for each of the 6 models. Note: compute the the train and test error for the typical threshold of $\theta = 0.5$.
- Among $C = 0.001, C = 0.1, C = 10$ which one leads to a simple model, and which to a complex model?
- **Use $C = 0.1$ model to do the following:**
- Find the 20 largest positive and negative estimated coeeficients, and print their corresponding words. What is your interptation of these words?

- Once the coeeficients of the logistic regression are estimated, we can use them predict the probability of a review being postive or negative. There is difference between reviews whose predicted probability is close to 0.5 and those reviews whose predicted probabilities are close to 1 or 0.

- Find the three reviews whose predicted probabilities is closest to 0.5. Then print the text of the review. These are reviews that are hardest to classify here.

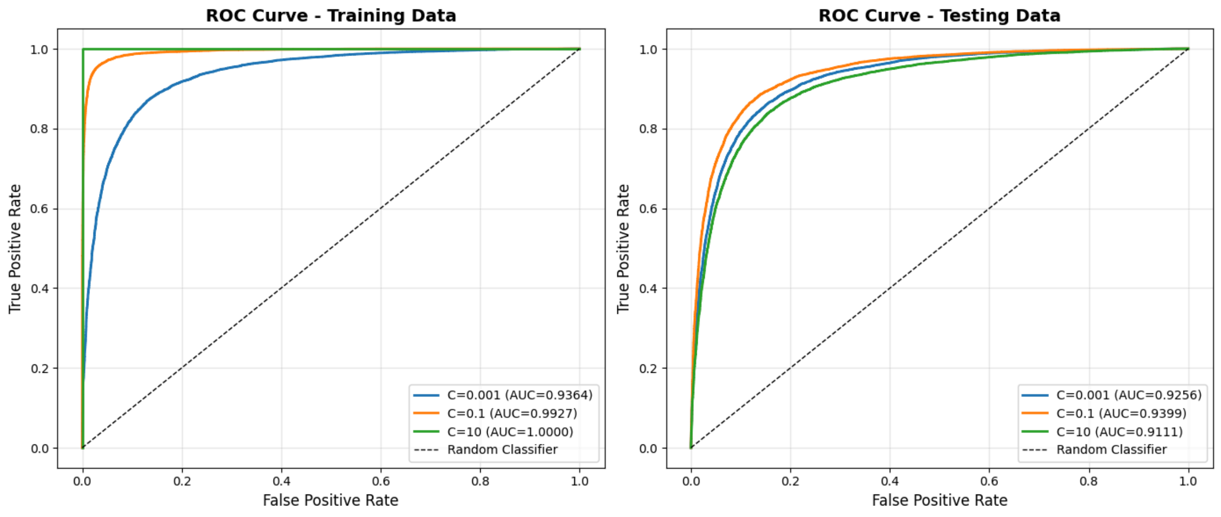
- Find the review whose predicted probability is closest to 1 or 0, respectivelty. These are reviews that seem easiest to classify.

In [8]:
```python
import tensorflow_datasets as tfds
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, roc_curve, auc
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import nltk
import re
import matplotlib.pyplot as plt

# Download NLTK data
nltk.download('stopwords')

# Download and Load IMDb dataset
dataset, info = tfds.load('imdb_reviews', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']

stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocess(text):
    text = text.decode('utf-8')
    text = re.sub(r'[^a-zA-Z\s]', ' ', text)
    text = text.lower()
    words = text.split()
    words = [stemmer.stem(word) for word in words if word not in stop_words]
    return ' '.join(words)

# Extract X and y from train and test datasets
X_train_list = [preprocess(text.numpy()) for text, _ in train_dataset]
y_train = np.array([label.numpy() for _, label in train_dataset])
X_test_list = [preprocess(text.numpy()) for text, _ in test_dataset]
y_test = np.array([label.numpy() for _, label in test_dataset])

# Print Train and Test sizes
print("Train Sample Size:", len(X_train_list))
print("Test Sample Size:", len(X_test_list))

# Vectorization of text data (TD matrix)
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train_list)
```

```
X_test = vectorizer.transform(X_test_list)

# Print shapes and statistics
n_train, p = X_train.shape
n_test = X_test.shape[0]
print(f"\nShape of X_train: {X_train.shape}")
print(f"Shape of X_test: {X_test.shape}")
print(f"n (training observations): {n_train}")
print(f"p (number of features): {p}")
print(f"Vocabulary Size: {len(vectorizer.vocabulary_)}")

# Ridge Logistic Regression for different C values
C_values = [0.001, 0.1, 10]
models = {}
train_errors = {}
test_errors = {}

print("\n" + "="*60)
print("Training Ridge Logistic Regression Models")
print("="*60)

for C in C_values:
    print(f"\nTraining model with C={C}...")
    model = LogisticRegression(penalty='l2', C=C, solver='liblinear', max_iter=1000
    model.fit(X_train, y_train)
    models[C] = model

    # Train error at threshold 0.5
    y_train_pred = model.predict(X_train)
    train_error = np.mean(y_train_pred != y_train)
    train_errors[C] = train_error

    # Test error at threshold 0.5
    y_test_pred = model.predict(X_test)
    test_error = np.mean(y_test_pred != y_test)
    test_errors[C] = test_error

    print(f"C={C}: Train Error={train_error:.4f}, Test Error={test_error:.4f}")

# Compute ROC curves and AUC for different thresholds
thresholds = np.arange(0.01, 1.0, 0.01)

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

for C in C_values:
    model = models[C]

    # Train ROC
    y_train_proba = model.predict_proba(X_train)[:, 1]
    fpr_train, tpr_train, _ = roc_curve(y_train, y_train_proba)
    auc_train = auc(fpr_train, tpr_train)
    axes[0].plot(fpr_train, tpr_train, label=f'C={C} (AUC={auc_train:.4f})', linewi

    # Test ROC
    y_test_proba = model.predict_proba(X_test)[:, 1]
    fpr_test, tpr_test, _ = roc_curve(y_test, y_test_proba)
```

```python
    auc_test = auc(fpr_test, tpr_test)
    axes[1].plot(fpr_test, tpr_test, label=f'C={C} (AUC={auc_test:.4f})', linewidth

# Format Train ROC plot
axes[0].plot([0, 1], [0, 1], 'k--', linewidth=1, label='Random Classifier')
axes[0].set_xlabel('False Positive Rate', fontsize=12)
axes[0].set_ylabel('True Positive Rate', fontsize=12)
axes[0].set_title('ROC Curve - Training Data', fontsize=14, fontweight='bold')
axes[0].legend(loc='lower right', fontsize=10)
axes[0].grid(True, alpha=0.3)

# Format Test ROC plot
axes[1].plot([0, 1], [0, 1], 'k--', linewidth=1, label='Random Classifier')
axes[1].set_xlabel('False Positive Rate', fontsize=12)
axes[1].set_ylabel('True Positive Rate', fontsize=12)
axes[1].set_title('ROC Curve - Testing Data', fontsize=14, fontweight='bold')
axes[1].legend(loc='lower right', fontsize=10)
axes[1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('roc_curves.png', dpi=300, bbox_inches='tight')
plt.show()

# Create error table
print("\n" + "="*60)
print("Train and Test Error Table (threshold=0.5)")
print("="*60)
print(f"{'C Value':<15} {'Train Error':<15} {'Test Error':<15}")
print("-"*45)
for C in C_values:
    print(f"{C:<15} {train_errors[C]:<15.4f} {test_errors[C]:<15.4f}")

print("\n" + "="*60)
print("Model Complexity Analysis")
print("="*60)
print(f"C=0.001: Strongest regularization → SIMPLEST model")
print(f"C=0.1:   Moderate regularization → MODERATE complexity")
print(f"C=10:    Weak regularization → MOST COMPLEX model")

# Analysis with C=0.1 model
print("\n" + "="*60)
print("Analysis with C=0.1 Model")
print("="*60)

model_01 = models[0.1]
coefficients = model_01.coef_[0]
feature_names = vectorizer.get_feature_names_out()

# Get indices of top 20 positive and negative coefficients
top_positive_idx = np.argsort(coefficients)[-20:][::-1]
top_negative_idx = np.argsort(coefficients)[:20]

print("\n20 Most Positive Coefficients (indicating positive sentiment):")
print("-" * 60)
for i, idx in enumerate(top_positive_idx, 1):
    print(f"{i:2d}. {feature_names[idx]:<20s} (coef: {coefficients[idx]:>8.4f})")
print("\n20 Most Negative Coefficients (indicating negative sentiment):")
print("-" * 60)
for i, idx in enumerate(top_negative_idx, 1):
    print(f"{i:2d}. {feature_names[idx]:<20s} (coef: {coefficients[idx]:>8.4f})")

print("\nInterpretation:")
print("Positive coefficients: Words like 'excel', 'perfect', 'best' strongly")
print("indicate positive reviews. These are words commonly used in praise.")
print("\nNegative coefficients: Words like 'worst', 'bad', 'wast' (waste) strongly"
print("indicate negative reviews. These express dissatisfaction and criticism.")

# Find reviews closest to 0.5 (hardest to classify)
y_test_proba = model_01.predict_proba(X_test)[:, 1]
distance_from_half = np.abs(y_test_proba - 0.5)
hardest_indices = np.argsort(distance_from_half)[:3]

print("\n" + "="*60)
print("3 Hardest Reviews to Classify (probability ≈ 0.5)")
print("="*60)

X_test_original = [text.numpy().decode('utf-8') for text, _ in test_dataset]
y_test_labels = [label.numpy() for _, label in test_dataset]

for i, idx in enumerate(hardest_indices, 1):
    print(f"\nReview {i} (Index: {idx}):")
    print(f"Predicted Probability: {y_test_proba[idx]:.4f}")
    print(f"True Label: {'Positive' if y_test_labels[idx] == 1 else 'Negative'}")
    print(f"Review Text (first 500 chars):")
    print("-" * 60)
    print(X_test_original[idx][:500] + "...")

# Find reviews closest to 0 and 1 (easiest to classify)
closest_to_zero_idx = np.argmin(y_test_proba)
closest_to_one_idx = np.argmax(y_test_proba)

print("\n" + "="*60)
print("Easiest Reviews to Classify")
print("="*60)

print(f"\nMost Confident NEGATIVE Review (probability ≈ 0):")
print(f"Index: {closest_to_zero_idx}")
print(f"Predicted Probability: {y_test_proba[closest_to_zero_idx]:.4f}")
print(f"True Label: {'Positive' if y_test_labels[closest_to_zero_idx] == 1 else 'Ne
print(f"Review Text (first 500 chars):")
print("-" * 60)
print(X_test_original[closest_to_zero_idx][:500] + "...")

print(f"\n\nMost Confident POSITIVE Review (probability ≈ 1):")
print(f"Index: {closest_to_one_idx}")
print(f"Predicted Probability: {y_test_proba[closest_to_one_idx]:.4f}")
print(f"True Label: {'Positive' if y_test_labels[closest_to_one_idx] == 1 else 'Neg
print(f"Review Text (first 500 chars):")
print("-" * 60)
print(X_test_original[closest_to_one_idx][:500] + "...")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\saad_\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Train Sample Size: 25000
Test Sample Size: 25000

Shape of X_train: (25000, 49642)
Shape of X_test: (25000, 49642)
n (training observations): 25000
p (number of features): 49642
Vocabulary Size: 49642

============================================================
Training Ridge Logistic Regression Models
============================================================

Training model with C=0.001...
C=0.001: Train Error=0.1318, Test Error=0.1456

Training model with C=0.1...
C=0.1: Train Error=0.0387, Test Error=0.1276

Training model with C=10...
C=10: Train Error=0.0002, Test Error=0.1594
```



```
============================================================
Train and Test Error Table (threshold=0.5)
============================================================
C Value         Train Error     Test Error
---------------------------------------------
0.001           0.1318          0.1456
0.1             0.0387          0.1276
10              0.0002          0.1594

============================================================
Model Complexity Analysis
============================================================
C=0.001: Strongest regularization → SIMPLEST model
C=0.1:   Moderate regularization → MODERATE complexity
C=10:    Weak regularization → MOST COMPLEX model

============================================================
Analysis with C=0.1 Model
============================================================

20 Most Positive Coefficients (indicating positive sentiment):
------------------------------------------------------------
 1. excel                (coef:   0.9040)
 2. refresh              (coef:   0.8801)
 3. superb               (coef:   0.7829)
 4. favorit              (coef:   0.7632)
 5. perfect              (coef:   0.7414)
 6. funniest             (coef:   0.7277)
 7. highli               (coef:   0.7057)
 8. surprisingli         (coef:   0.6747)
 9. perfectli            (coef:   0.6694)
10. fantast              (coef:   0.6621)
11. delight              (coef:   0.6478)
12. flawless             (coef:   0.6296)
13. brilliant            (coef:   0.6145)
14. gem                  (coef:   0.5942)
15. subtl                (coef:   0.5782)
16. amaz                 (coef:   0.5554)
17. sensit               (coef:   0.5514)
18. today                (coef:   0.5469)
19. hilari               (coef:   0.5283)
20. enjoy                (coef:   0.5277)

20 Most Negative Coefficients (indicating negative sentiment):
------------------------------------------------------------
 1. worst                (coef:  -1.3572)
 2. wast                 (coef:  -1.2287)
 3. aw                   (coef:  -1.1272)
 4. poorli               (coef:  -1.0589)
 5. bore                 (coef:  -0.8388)
 6. wors                 (coef:  -0.8363)
 7. disappoint           (coef:  -0.7863)
 8. dull                 (coef:  -0.7678)
 9. badli                (coef:  -0.7441)
10. horribl              (coef:  -0.7402)
11. embarrass            (coef:  -0.7197)
```

```
12. insult              (coef:  -0.7006)
13. poor                (coef:  -0.6998)
14. mess                (coef:  -0.6939)
15. unless              (coef:  -0.6890)
16. mst                 (coef:  -0.6881)
17. unfunni             (coef:  -0.6876)
18. unfortun            (coef:  -0.6828)
19. alright             (coef:  -0.6823)
20. forgett             (coef:  -0.6795)

Interpretation:
Positive coefficients: Words like 'excel', 'perfect', 'best' strongly
indicate positive reviews. These are words commonly used in praise.

Negative coefficients: Words like 'worst', 'bad', 'wast' (waste) strongly
indicate negative reviews. These express dissatisfaction and criticism.

============================================================
3 Hardest Reviews to Classify (probability ≈ 0.5)
============================================================

Review 1 (Index: 4754):
Predicted Probability: 0.4998
True Label: Negative
Review Text (first 500 chars):
------------------------------------------------------------
This is a flic that you can safely avoid. childish plot and poor acting from Cameron
(she really shouldn't take on these serious roles - they don't suit her)Besides she
looks most unglamorous, Lets face it, shes' great in Charlies Angels and She was sup
erb in There's Something about Mary... The Pope wouldn't be a good President of the
USA and vice versa, my point being that there are some actors/actresses who are suit
ed to a particular type of character and Cameron is most unsuited for this seri...

Review 2 (Index: 2021):
Predicted Probability: 0.5002
True Label: Negative
Review Text (first 500 chars):
------------------------------------------------------------
Ben, (Rupert Grint), is a deeply unhappy adolescent, the son of his unhappily marrie
d parents. His father, (Nicholas Farrell), is a vicar and his mother, (Laura Linne
y), is ... well, let's just say she's a somewhat hypocritical soldier in Jesus' arm
y. It's only when he takes a summer job as an assistant to a foul-mouthed, eccentri
c, once-famous and now-forgotten actress Evie Walton, (Julie Walters), that he final
ly finds himself in true 'Harold and Maude' fashion. Of course, Evie is deeply unha
p...

Review 3 (Index: 12893):
Predicted Probability: 0.4997
True Label: Negative
Review Text (first 500 chars):
------------------------------------------------------------
To some of us, director Ernst Lubitsch, adored for his underlying cheekiness and iro
nic comic touches, was rather wet when it came to picking material. It isn't that Lu
bitsch is overrated--on the contrary, he probably was ahead of his time in terms of
a visual narrative--yet the projects he became attached to (or was assigned to) are
not quite the landmarks of comedy his fans like to label them. With "Heaven Can Wai
```

```
t", a screen-adaptation of Lazlo Bus-Fekete's play "Birthday", Lubitsch is saddle...

============================================================
Easiest Reviews to Classify
============================================================

Most Confident NEGATIVE Review (probability ≈ 0):
Index: 18499
Predicted Probability: 0.0000
True Label: Negative
Review Text (first 500 chars):
------------------------------------------------------------
Rose � Does anything actually happen in this episode? It introduces our two leads, a
slow-witted grinning idiot of a Doctor and an utterly un-interesting companion. Ther
e's no plot to speak of, childish humour, mixed with some extremely bad pacing and i
ncidental music. What else is there to say, really?<br /><br />The End of the World
� A marginal improvement, in that we see our first outer-space scenario. Subsequentl
y brought down by poor contemporary humour, paper-thin logic, very poor pacing,...


Most Confident POSITIVE Review (probability ≈ 1):
Index: 10792
Predicted Probability: 1.0000
True Label: Positive
Review Text (first 500 chars):
------------------------------------------------------------
Tony Hawk's Pro Skater 2x, isn't much different at all from the previous games (excl
uding Tony Hawk 3). The only thing new that is featured in Tony Hawk's Pro Skater 2
x, is the new selection of levels, and tweaked out graphics. Tony Hawk's Pro Skater
2x offers a new career mode, and that is the 2x career. The 2x career is basically T
ony Hawk 1 career, because there is only about five challenges per level. If you mis
sed Tony Hawk 1 and 2, I suggest that you buy Tony Hawk's Pro Skater 2x, but if y...
```