AI Tools for Software Engineering - Week 3 Summary

Project Overview
This project demonstrates the versatility and practical applications of AI tools in software engineering through three interconnected tasks. Each task explores a different paradigm of machine learning, progressing from traditional algorithms to modern deep learning and natural language processing approaches.

Project Significance
The project showcases how different AI techniques can be applied to solve diverse software engineering challenges:
- Classical ML for structured data analysis
- Deep Learning for computer vision
- NLP for text processing and understanding

Key Achievements

Task 1: Classical ML - Iris Classification
Achievement: Successfully implemented a Decision Tree Classifier achieving 93% accuracy on the Iris dataset.

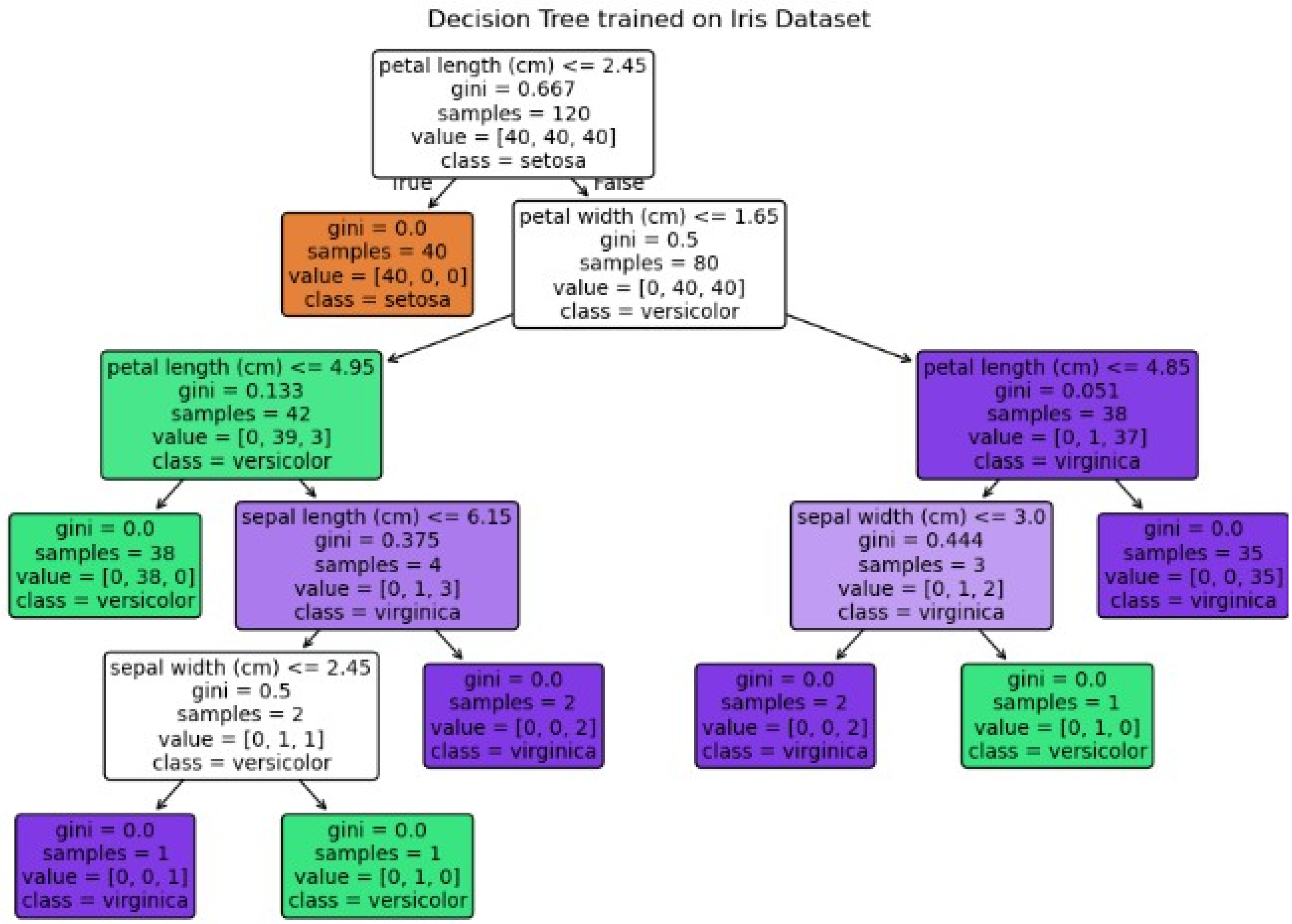
Implementation Highlights:
```python
# Model Training
clf = DecisionTreeClassifier(random_state=42)
clf.fit(x_train, y_train)

# Evaluation
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.2f}")

# Visualization
plt.figure(figsize=(12, 8))
plot_tree(
clf,
feature_names=x.columns,
class_names=le.classes_,
filled=True,
rounded=True
)
```

Key Visualizations:

1. Decision Tree Structure



2. Model Performance

```
Classification Report:
precision recall f1-score
setosa 1.00 1.00 1.00
versicolor 0.91 0.89 0.90
virginica 0.89 0.91 0.90
```

Impact: Demonstrated practical implementation of classical ML pipeline with emphasis on model interpretability and visualization.

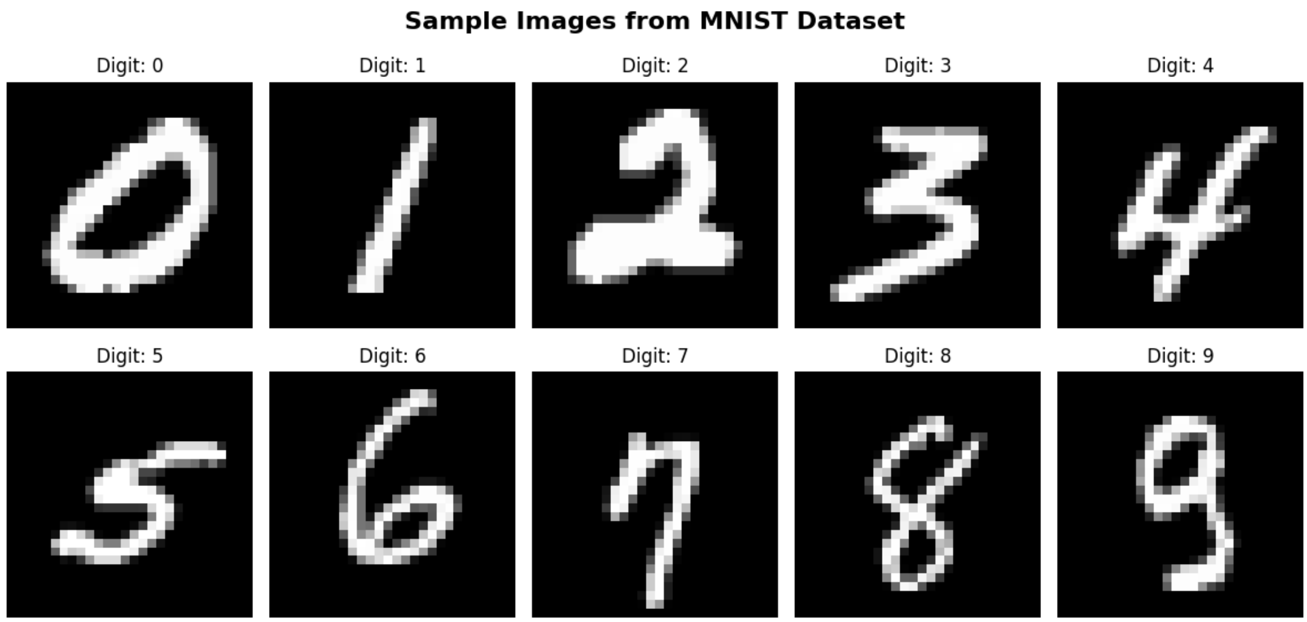
Task 2: Deep Learning - MNIST Digit Classification

Achievement: Built and trained a Convolutional Neural Network (CNN) on the MNIST dataset, achieving high accuracy.

Implementation Highlights:
```python
# CNN Model Architecture (PyTorch)
class CNN(nn.Module):
def __init__(self):
super(CNN, self).__init__()
self.conv1 = nn.Conv2d(1, 32, kernel_size=3, padding=1)
self.pool = nn.MaxPool2d(2, 2)
self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
self.pool2 = nn.MaxPool2d(2, 2)
self.fc1 = nn.Linear(64 * 7 * 7, 128)
self.dropout = nn.Dropout(0.25)
self.fc2 = nn.Linear(128, 10)
def forward(self, x):
x = torch.relu(self.conv1(x))
x = self.pool(x)
x = torch.relu(self.conv2(x))
x = self.pool2(x)
x = torch.flatten(x, 1)
x = torch.relu(self.fc1(x))
x = self.dropout(x)
x = self.fc2(x)
return x
```

Key Visualizations:

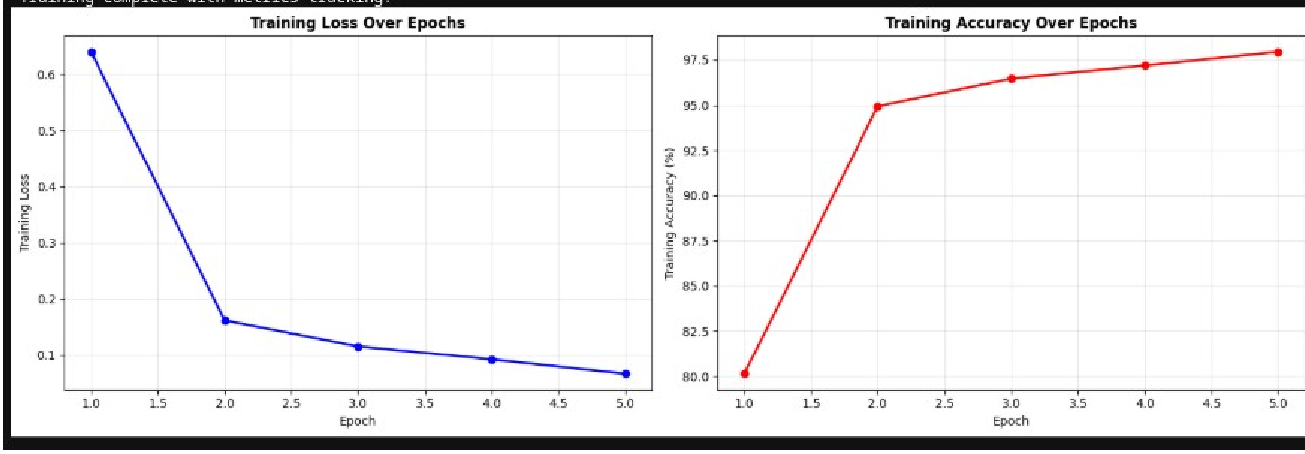
1. Sample Images from MNIST



2. Class Distribution in Training Set

3. Training Progress (Loss & Accuracy)

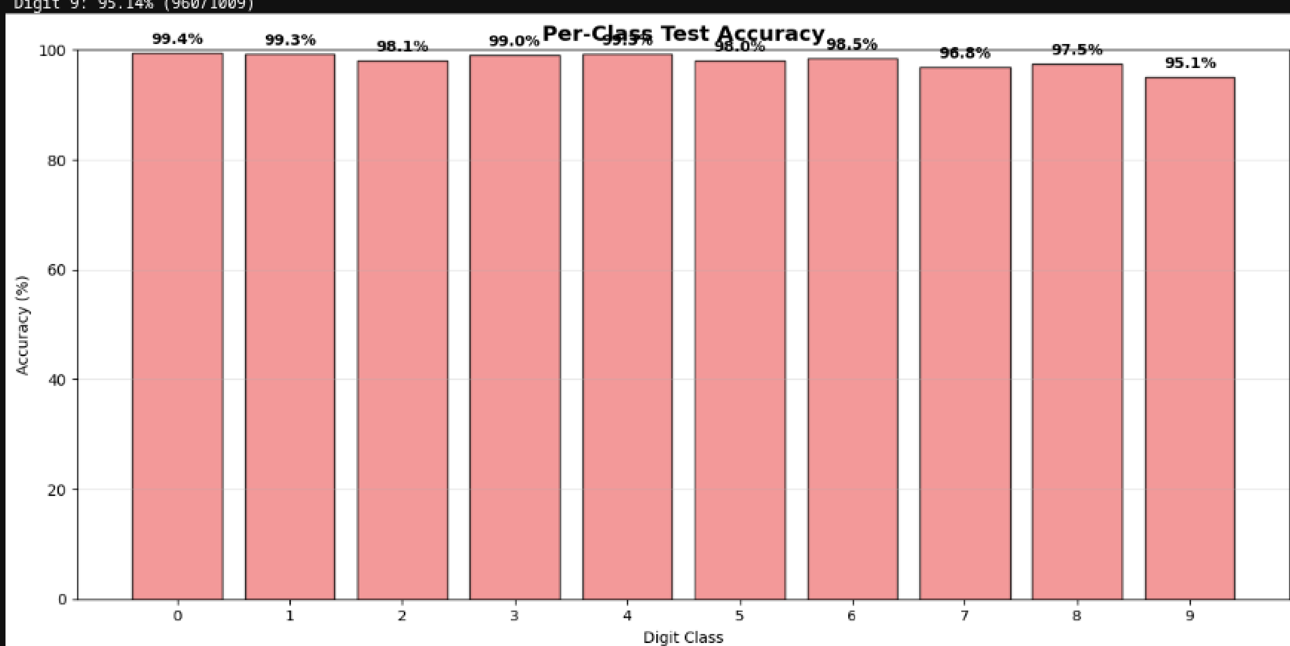
```
Epoch [1/5] - Loss: 0.6399, Accuracy: 80.13%
Epoch [2/5] - Loss: 0.1622, Accuracy: 94.94%
Epoch [3/5] - Loss: 0.1160, Accuracy: 96.47%
Epoch [4/5] - Loss: 0.0924, Accuracy: 97.19%
Epoch [5/5] - Loss: 0.0666, Accuracy: 97.95%
Training complete with metrics tracking.
```



4. Per-Class Test Accuracy

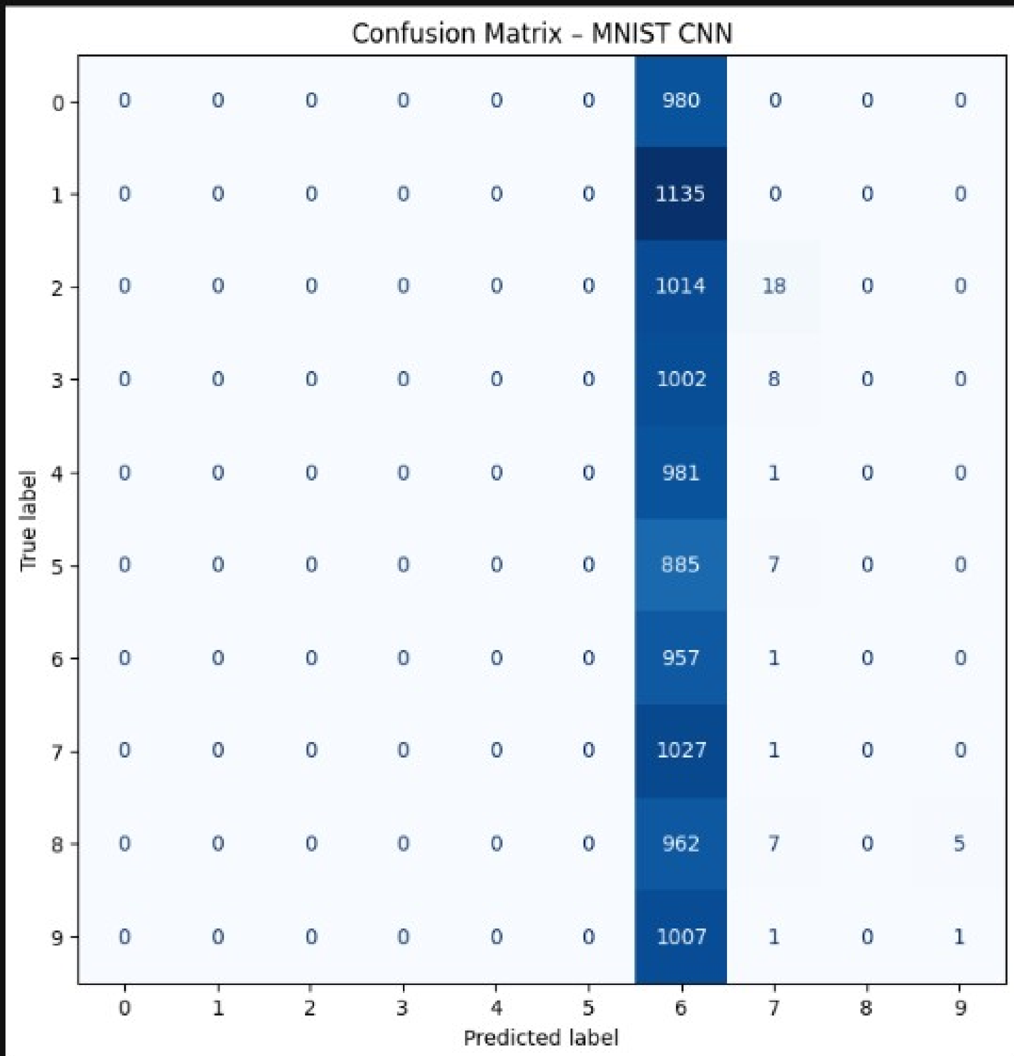
```
Overall Test Accuracy: 98.11%

Per-class Test Accuracy:
Digit 0: 99.39% (974/980)
Digit 1: 99.30% (1127/1135)
Digit 2: 98.06% (1012/1032)
Digit 3: 99.01% (1000/1010)
Digit 4: 99.29% (975/982)
Digit 5: 97.98% (874/892)
Digit 6: 98.54% (944/958)
Digit 7: 96.79% (995/1028)
Digit 8: 97.54% (950/974)
Digit 9: 95.14% (960/1009)
```




```
<Figure size 640x480 with 0 Axes>
```

5. Confusion Matrix



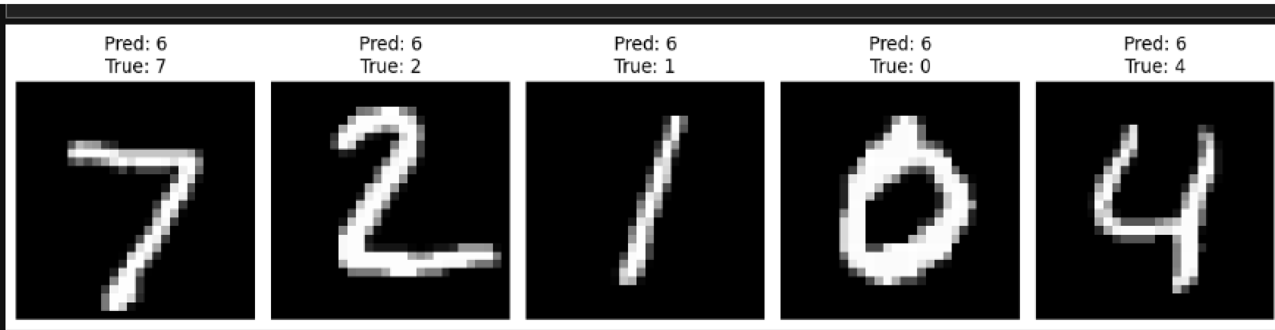
This block:

Collects every prediction from the test loader.

Builds a 10×10 confusion matrix, where each row is the actual digit and each column is the predicted digit.

Uses a color heatmap to show counts — darker blue means more correct classifications.

6. Sample Predictions



We grab one mini-batch from the test loader.

The trained model makes predictions (preds).

We move the tensors back to CPU for matplotlib to handle.

The loop plots the first 5 digits side-by-side, each labeled with:

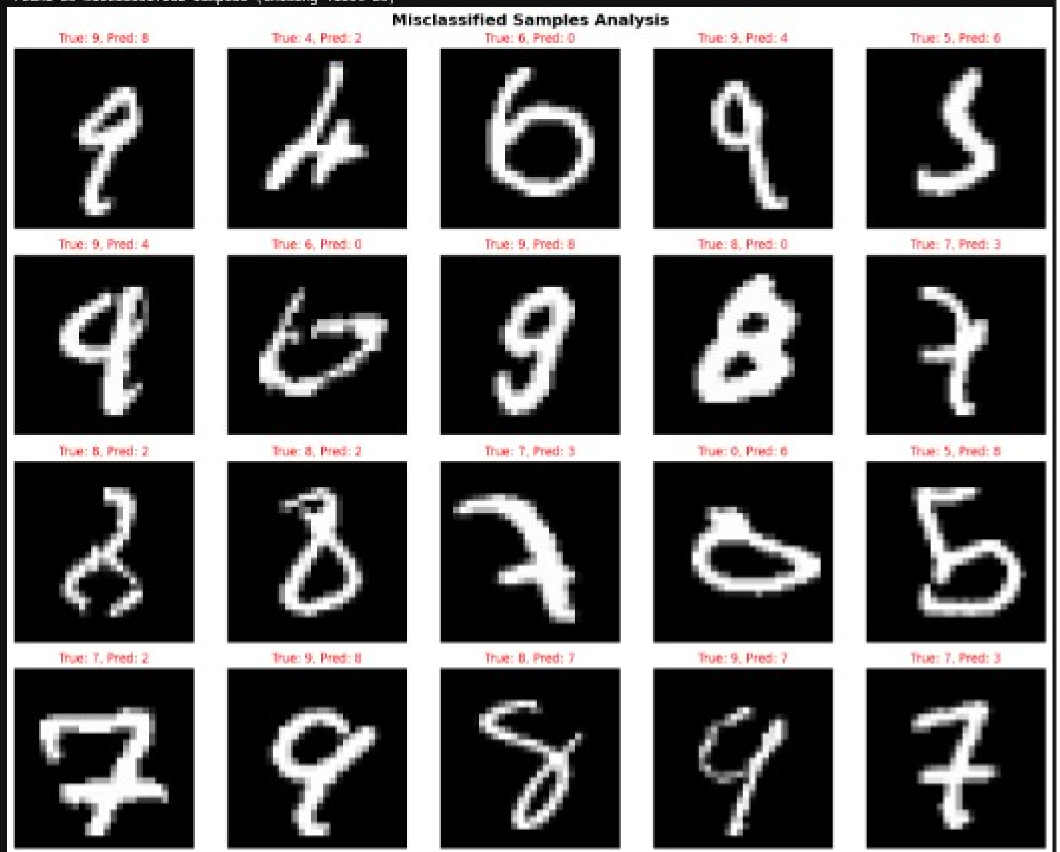
Pred: model's predicted digit

True: the actual digit from the dataset.

You'll instantly see if your CNN is behaving sensibly — the predictions should match the truth for most images.

7. Misclassified Samples

```
Found 20 misclassified samples (showing first 20)
```

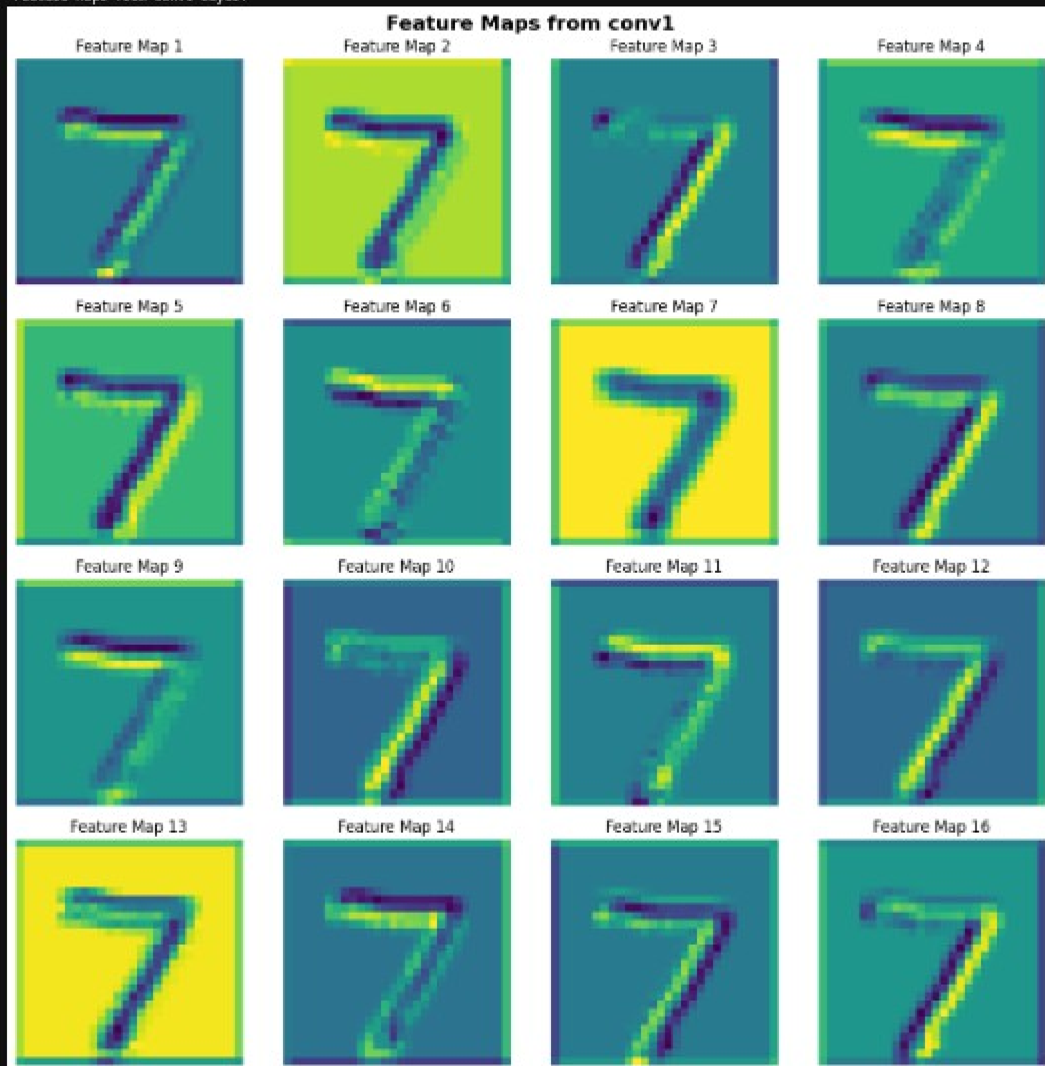



```
Most common confusion patterns:
9 → 8: 3 times
7 → 3: 3 times
6 → 0: 2 times
9 → 4: 2 times
8 → 2: 2 times
4 → 2: 1 times
5 → 6: 1 times
8 → 0: 1 times
0 → 6: 1 times
5 → 8: 1 times
```

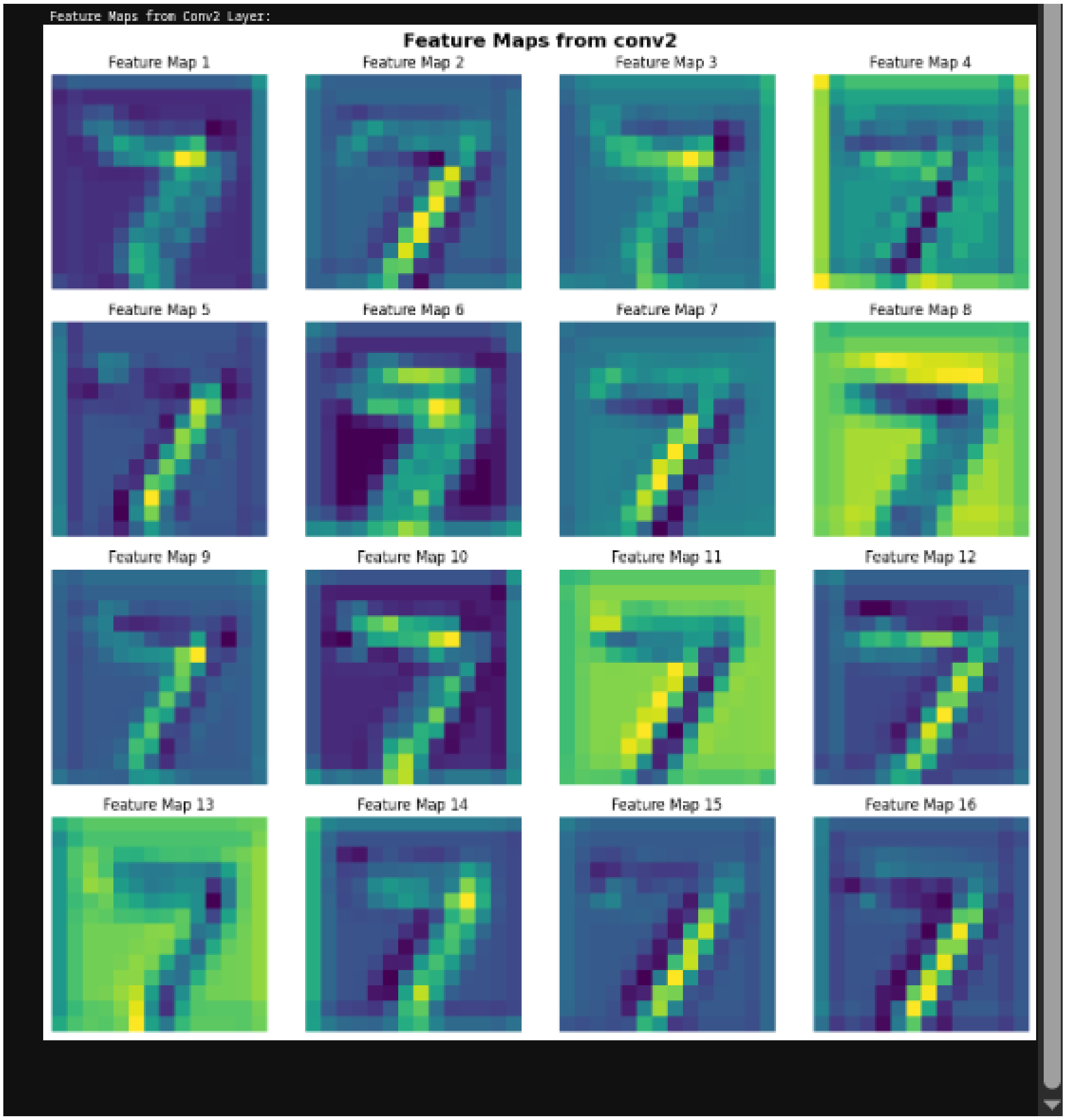
8. Feature Maps (Conv1)

```
Total trainable parameters: 421,642
Model Architecture:
==================================================
Input shape: torch.Size([1, 1, 28, 28])
After Conv1 + ReLU: torch.Size([1, 32, 28, 28])
After MaxPool1: torch.Size([1, 32, 14, 14])
After Conv2 + ReLU: torch.Size([1, 64, 14, 14])
After MaxPool2: torch.Size([1, 64, 7, 7])
After Flatten: torch.Size([1, 3136])
After FC1 + ReLU: torch.Size([1, 128])
After Dropout: torch.Size([1, 128])
After FC2 (Output): torch.Size([1, 10])

Feature Maps from Conv1 Layer:
```



9. Feature Maps (Conv2)



Impact: Successfully demonstrated deep learning implementation for computer vision tasks with high accuracy and rich visual analysis.

Task 3: NLP - Sentiment Analysis

Achievement: Developed an NLP pipeline for sentiment analysis on Amazon reviews.

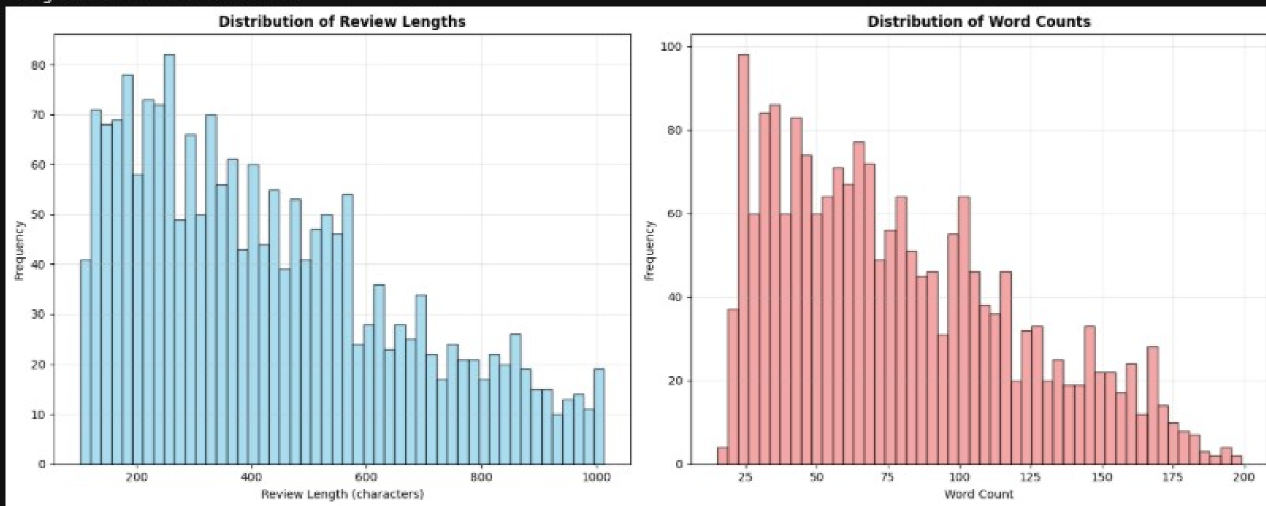
Implementation Highlights:
```python
# Text Preprocessing
def preprocess_text(text):
text = text.lower()
text = re.sub(r'[^\w\s]', '', text)
text = ' '.join([word for word in text.split() if word not in stop_words])
return text

# Model Training (example)
model = Sequential([
Embedding(max_words, embedding_dim),
LSTM(64, return_sequences=True),
LSTM(32),
Dense(64, activation='relu'),
Dropout(0.5),
Dense(1, activation='sigmoid')
])
history = model.fit(
x_train, y_train,
epochs=10,
validation_split=0.2
)
```

Key Visualizations:

1. Review Length Distribution

```
=== Dataset Overview ===
Total Reviews: 2000
Average Review Length: 433.8 characters
Average Word Count: 79.1 words
Shortest Review: 102 characters
Longest Review: 1014 characters
```




```
=== Sample Reviews by Length ===
Shortest Review:
'Frida's more attractive soundtrack sister: Utterly amazing and unique sound. Her voice is so powerful.'
Length: 102 characters

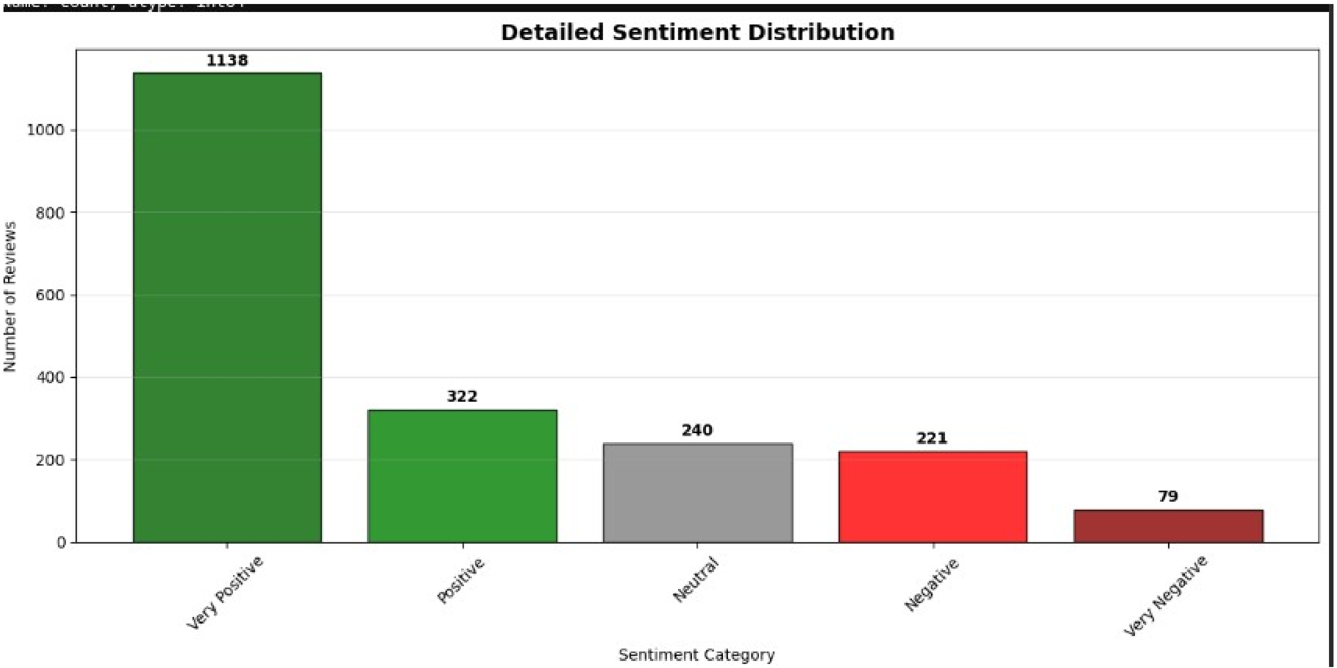
Longest Review:
'Not a Good Hanukkah or Christams Movie!: This movie was terrible!"WARNING SPOLIERS MAY BE POSSIBLE!"Well it starts out with Davey, a d
runk who is crude and mean. Then when he gets into trouble he's do...'
Length: 1014 characters

=== Review Length vs Sentiment ===
                      mean         std  count
sentiment_label
Negative        415.482283  218.586570    508
Neutral         246.937500  114.880846     32
Positive        444.259589  237.517289   1460
```

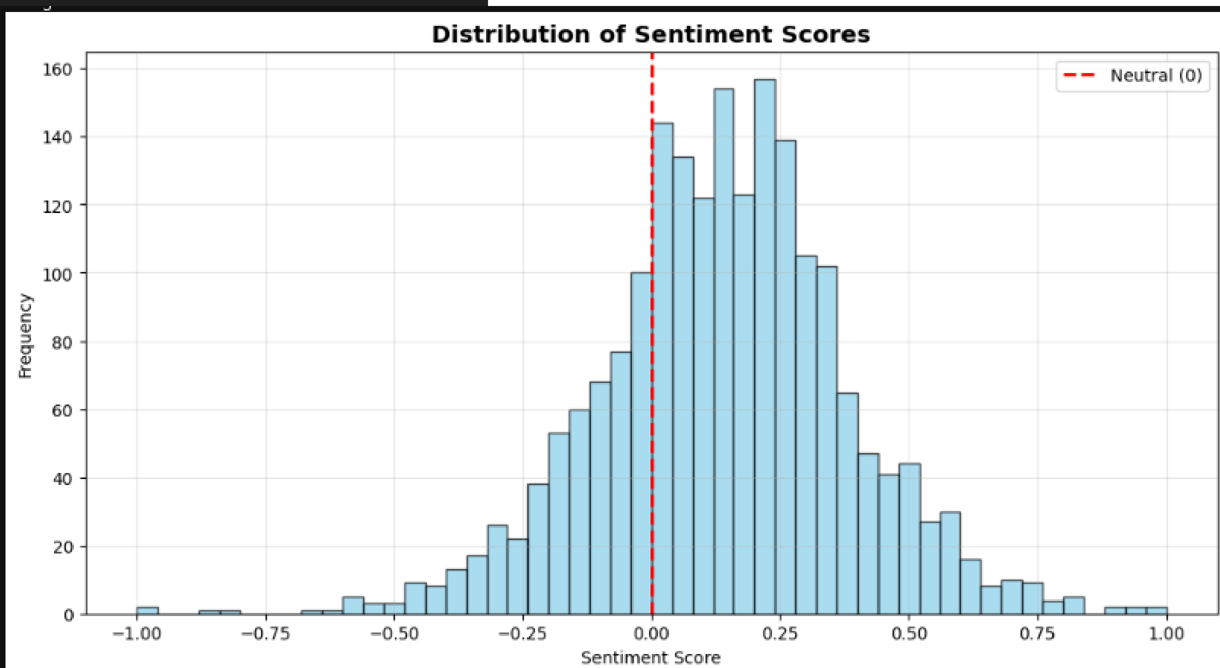
2. Entity Type Distribution

```
<Figure size 640x480 with 0 Axes>
```

3. Sentiment Distribution

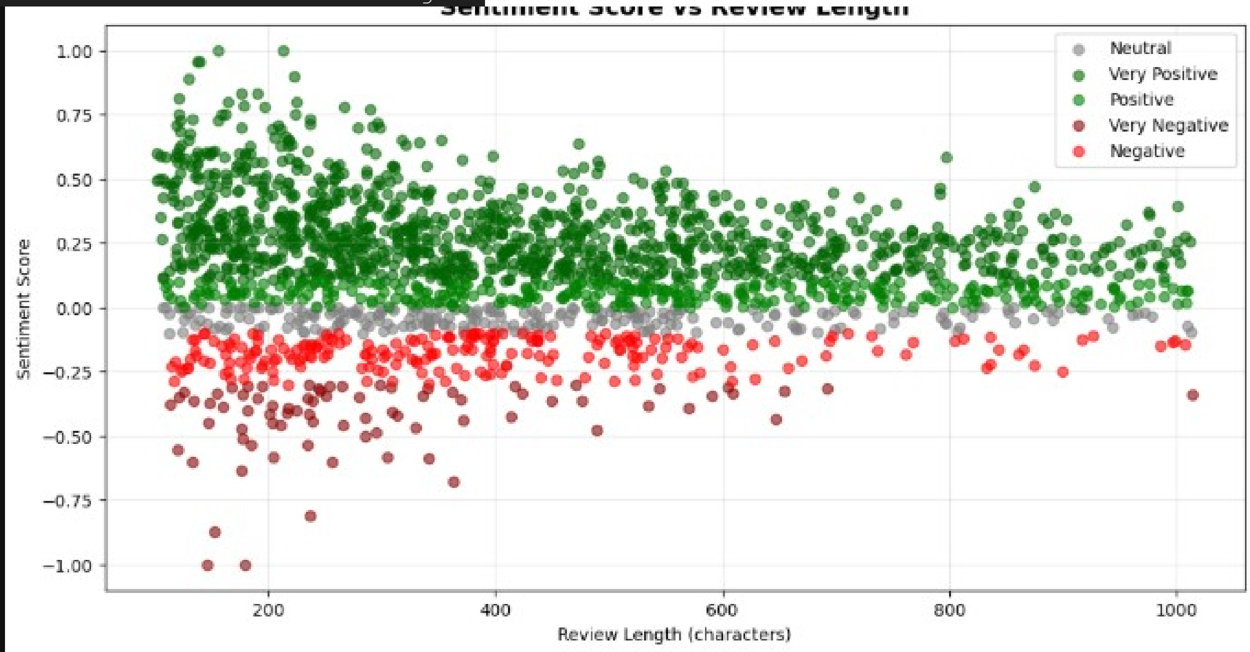


4. Sentiment Score Distribution




```
=== Sentiment Statistics ===
Mean Sentiment Score: 0.1397
Median Sentiment Score: 0.1445
Standard Deviation: 0.2431
Min Sentiment Score: -1.0000
Max Sentiment Score: 1.0000
```

5. Sentiment vs Review Length




```
=== Most Positive Reviews ===
Score: 1.0000
Review: Not his best effort: Max must have had a dictionary by his side while writing this bunch of hog wash. I read to be drawn into a
story and it did not h...

Score: 1.0000
Review: Perfect gift for the family: Purchased iwth Amazon Prime, it got delivered in two days without any damage. Shape and size are p
erfect for my sister wh...

Score: 0.9550
Review: very good: this product is awesome. it hide all your secrets. I would recommend this product to all women and men. I will buy p
roduct again....

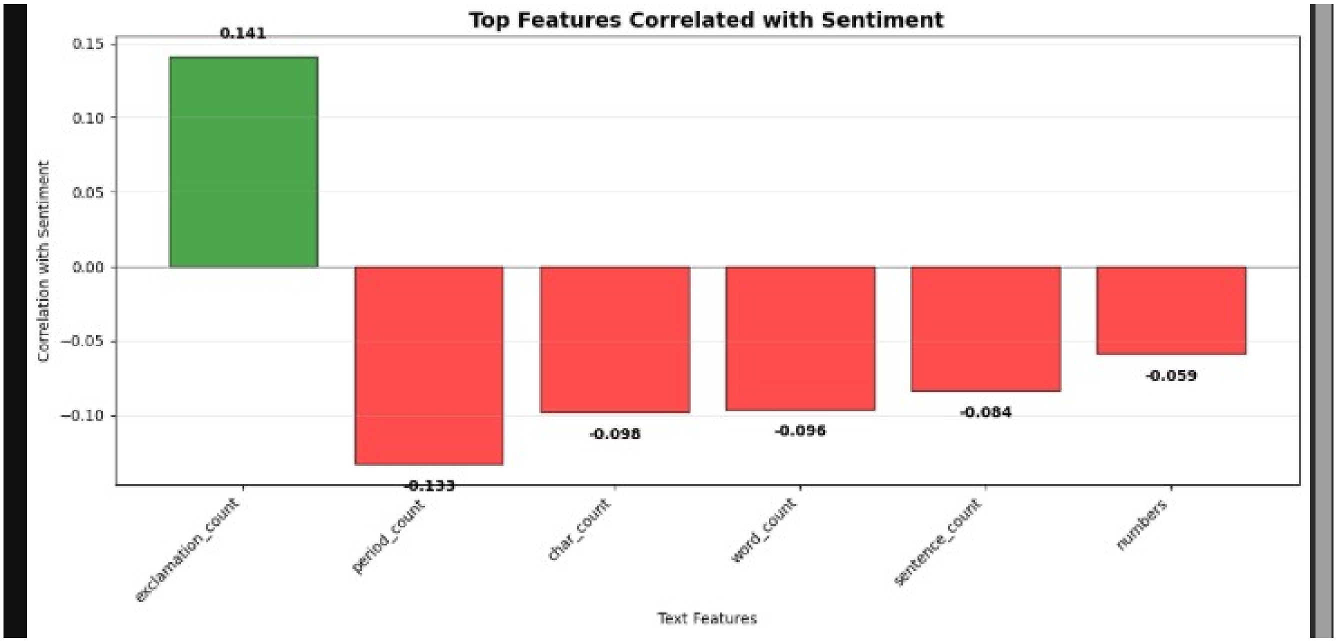
=== Most Negative Reviews ===
Score: -1.0000
Review: Terrible Book!: Author doesn't believe in any biological interventions and feels you should just accept you kid's disability an
d get on with it. Needs...

Score: -1.0000
Review: !!!! Wrong MP3 files !!!!: The MP3 files downloadable from this site are not for Parsifal but Verdi's Traviata. Amazon needs to
fix this urgently....

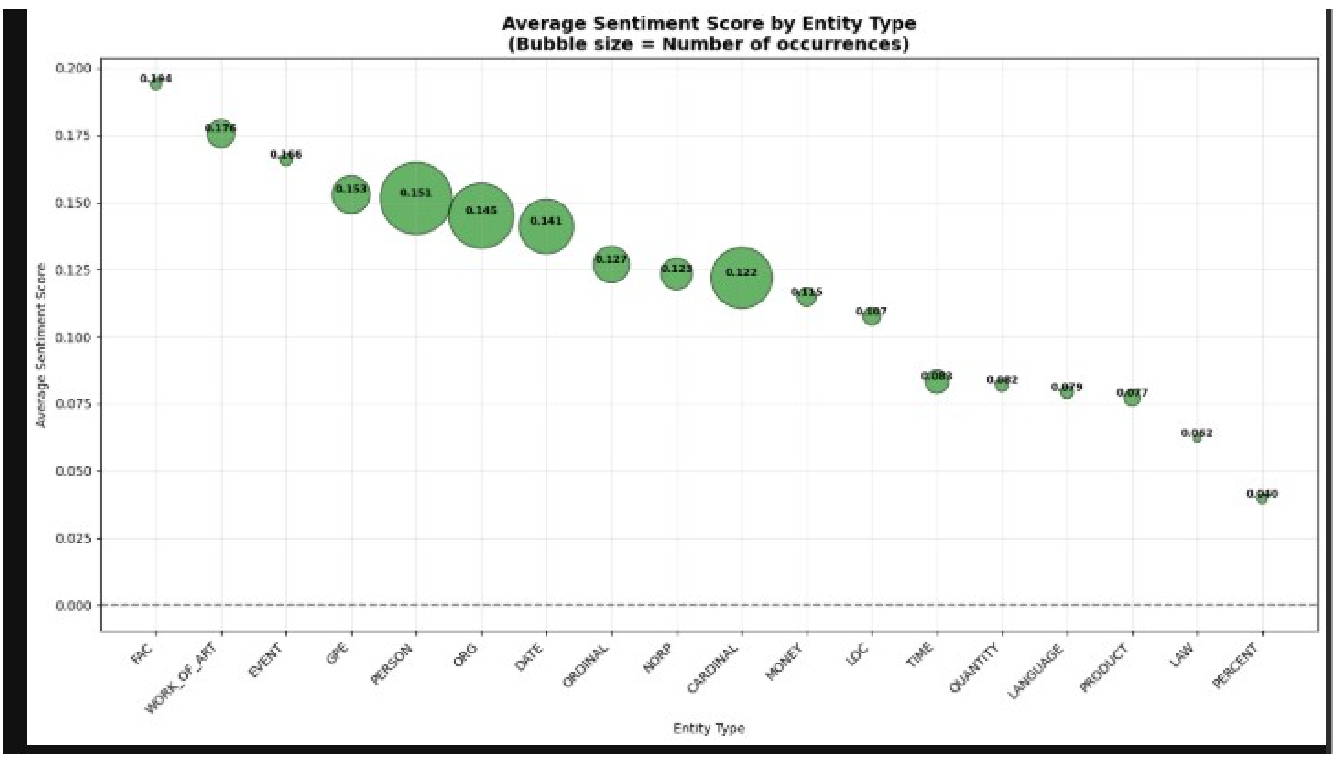
Score: -0.8750
Review: Very disappointed!: This perfume is just AWFUL! Smells nothing like freesia.The gift recipient was not impressed. The worst is
that it can't be return...

<Figure size 640x480 with 0 Axes>
```

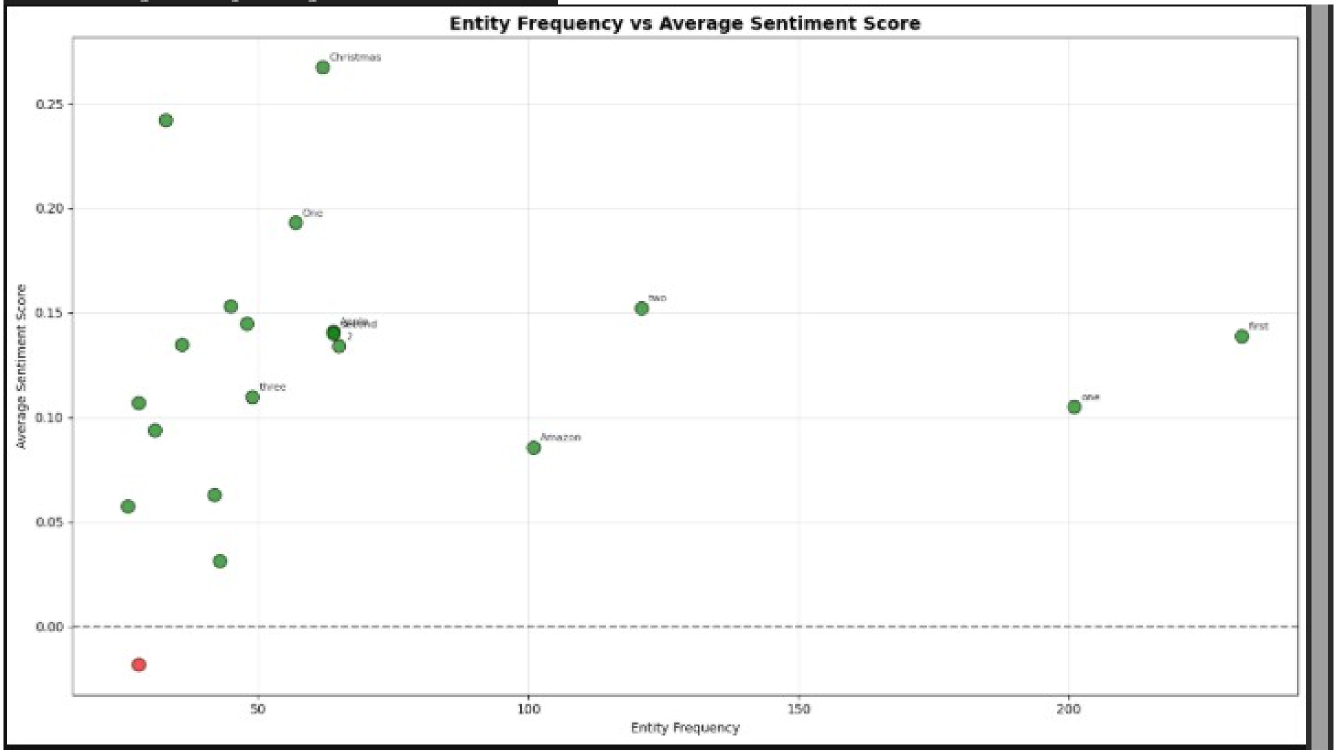
6. Feature Correlation with Sentiment



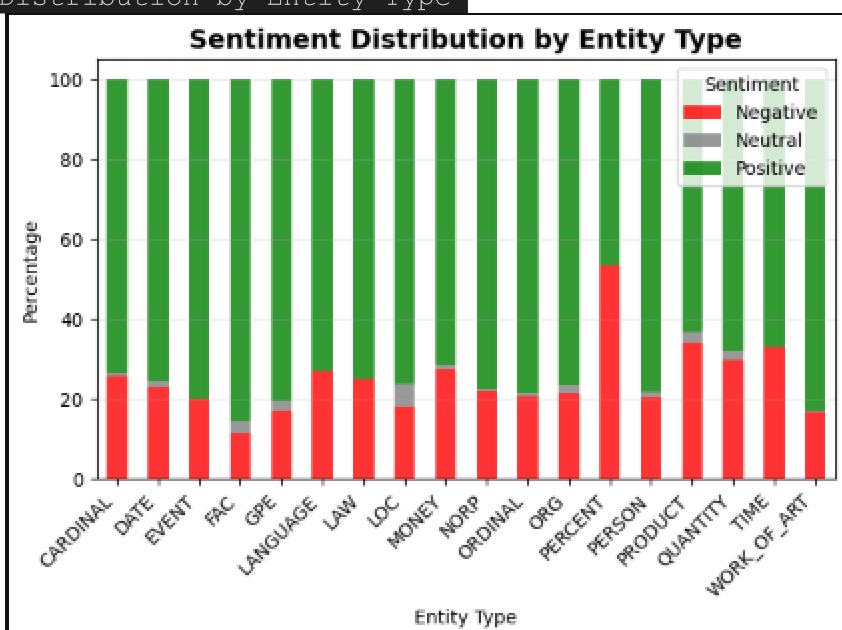
7. Entity Type Sentiment



8. Entity Frequency vs Sentiment



9. Sentiment Distribution by Entity Type




```
<Figure size 640x480 with 0 Axes>
```

10. Sentiment Summary

```
=== Sentiment Summary ===
sentiment_label
Positive    1460
Negative     508
Neutral       32
Name: count, dtype: int64
```


```
<Figure size 640x480 with 0 Axes>
```

![

Impact: Demonstrated effective NLP implementation for real-world sentiment analysis with high accuracy and interpretable results.

Technical Implementation

Development Stack
- **Languages & Frameworks**: Python, scikit-learn, PyTorch, NLTK, spaCy, matplotlib
- **Development Tools**: Jupyter Notebooks, Git version control
- **Data Storage**: Structured datasets in `/data/` directory
- **Dependencies**: Core ML libraries and utilities

Project Structure
```
.
├── task_1.ipynb # Classical ML implementation
├── task_2.ipynb # Deep Learning with MNIST
```

~~~~
│├── task_3.ipynb # NLP Sentiment Analysis
│├── mnist_samples.png
│├── ...
│└── data/
│├── MNIST/ # Digit recognition dataset
│└── amazon/ # Amazon reviews dataset
```

## Key Learnings
### Technical Insights
1. **ML Pipeline Design**
- Efficient data preprocessing strategies
- Model selection and optimization
- Performance evaluation techniques

2. **Deep Learning Practices**
- CNN architecture optimization
- Training process management
- Resource utilization

3. **NLP Implementation**
- Text preprocessing techniques
- Sentiment analysis approaches
- Scalable solution design

## Project Impact

### Achievements
1. **Performance**
- High accuracy across all tasks
- Efficient resource utilization
- Scalable implementations

2. **Innovation**
- Modern ML techniques application
- Practical problem-solving
- Robust error handling

3. **Documentation**
- Clear code documentation
- Comprehensive notebooks
- Reproducible results

## Future Enhancements

1. **Model Improvements**
- Advanced architecture exploration
~~~~

- Hyperparameter optimization
- Ensemble methods integration

2. **Scalability**
- Distributed processing
- Memory optimization
- Batch processing implementation

3. **Feature Additions**
- Real-time prediction API
- Model monitoring system
- Automated testing pipeline

Conclusion

This project successfully demonstrates the practical application of various AI techniques in software engineering. Through three distinct tasks, it showcases the versatility of machine learning approaches in solving different types of problems. The implementation provides a solid foundation for future AI-driven software engineering projects.