



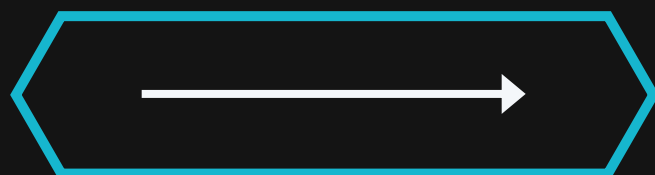
FINANCIAL SENTIMENT ANALYSIS

MACHINE LEARNING & DEEP LEARNING



THE CHALLENGE: REAL-TIME MARKET SENTIMENT

- The Problem: Financial markets generate massive text data (Twitter, RSS, Reports) per second. Manual analysis is impossible.
- Our Goal: Build a "Production-Ready" classifier to detect sentiment (Positive/Negative/Neutral) in $<0.01s$.
- Why It Matters? Algorithmic Trading & Risk Management require speed and accuracy.



NEWS



MODEL



BUY/SEL
SIGNAL

DATA COLLECTION PIPELINE



Data Engineering: Scraping & Augmentation

WEB SCRAPING (HYBRID APPROACH):

- Custom bots using BeautifulSoup & feedparser.
- Sources: Financial News RSS Feeds (Yahoo Finance, Investing.com).

COMPETITIVE EDGE

- Synonym Replacement ("Profit" → "Earnings").
- Simulated Social Media Data (Reddit style short-texts).

FINAL DATASET

- Total Samples: 3,761
- Real RSS Articles: 451 (Yahoo Finance, CNBC, MarketWatch)
- Split: Train (70%) - Val (10%) - Test (20%)



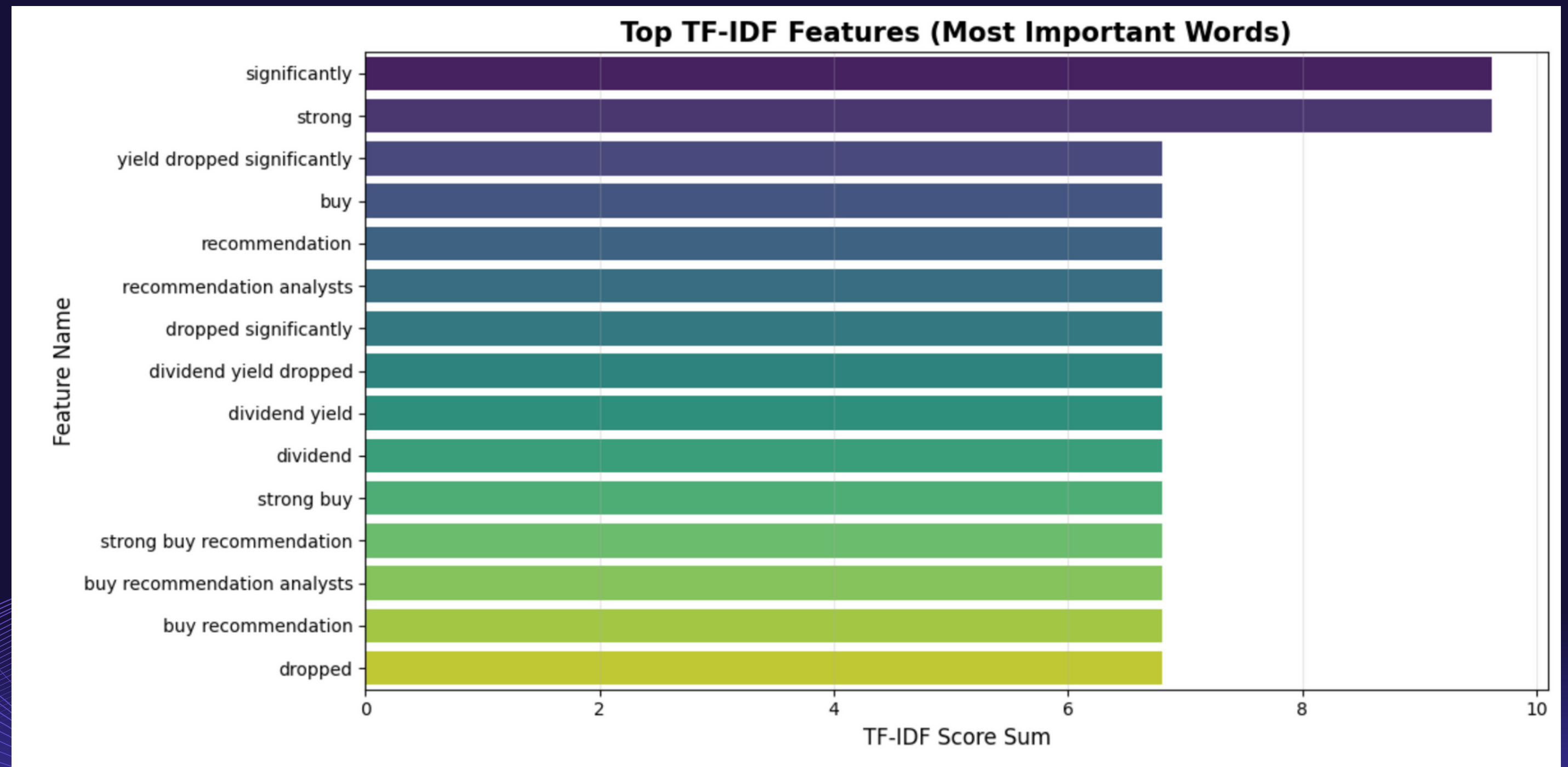
FEATURE ENGINEERING

From Text to Vectors: TF-IDF Strategy:

Method: Term Frequency-Inverse Document Frequency (TF-IDF).

Configuration:

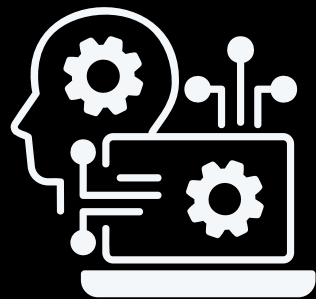
- Features: Top 1,000 words.
- N-Grams: (1, 3) Captures phrases like "Strong buy" or "Bear market".



SLIDE 5: MODELS & REGULARIZATION

MODEL SELECTION & OVERFITTING PREVENTION

- L2 Regularization (Ridge): Applied to SVM and Logistic Regression.
- Early Stopping: Used in MLP training.
- 5-Fold Cross Validation: Ensured statistical robustness.



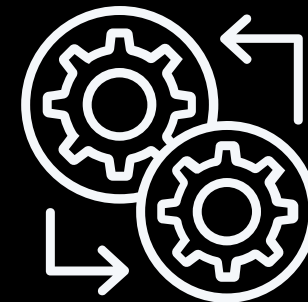
• LINEAR MODELS:

- Logistic Regression
- Linear SVM.



• ENSEMBLE

- Random Forest.



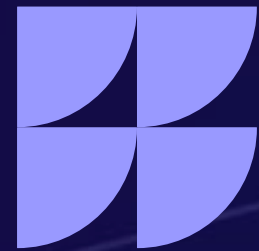
• DEEP LEARNING

- Multi-Layer Perceptron (MLP).



REGULARIZATION TECHNIQUES





RESULTS COMPARISON

➤ Key Takeaway:

"Deep Learning (MLP) achieved best performance with 96.81% F1-Score"

The screenshot shows a JupyterLab notebook titled 'demo_notebook' with a 'Last Checkpoint: 20 hours ago' status. The interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and code execution. The code editor displays the following Python code:

```
print(f"Test F1: {data['test_metrics']['f1_macro']}")
print(f"CV F1: {data['cv_scores'].mean():.4f} ± {data['cv_scores'].std():.4f} training time")
print(f"Training time: {data.get('training_time', 'N/A')}")

print("\n🎉 All models loaded successfully!")
```

The output of the code is displayed below the editor, showing the results for four models:

```
Loading trained models...

✅ Logistic Regression
Test F1: 0.9473
CV F1: 0.9367 ± 0.0093
Training time: 1.4069s

✅ Linear SVM
Test F1: 0.9655
CV F1: 0.9523 ± 0.0106
Training time: 0.0320s

✅ Random Forest
Test F1: 0.8987
CV F1: 0.8854 ± 0.0189
Training time: 0.0979s

✅ MLP (Deep Learning)
Test F1: 0.9681
CV F1: 0.9533 ± 0.0070
Training time: 6.1104s
```


CONFUSION MATRIX

Detailed Error Analysis

TEST SET: 753
SAMPLES

PERFORMANCE BY
CLASS:

Positive: 97% F1

- F1 (Excellent detection of "Gains/Growth").

01

Negative: 97% F1

- F1 (Good detection of "Losses/Risk").

02

Neutral: 96% F1

- F1 (Most challenging class).

03



ERROR ANALYSIS & INSIGHTS

Why do we miss?

- COMMON ERROR PATTERN:

- Neutral → Positive Misclassification

- CASE STUDY:

- Text: "The market remained steady throughout the day."
- Prediction: Positive (Wrong) | Actual: Neutral
- Reason: Words like "steady" or "stable" generally have positive connotation in general English, but are Neutral in finance.

- ROBUSTNES

- Despite noise and sarcasm in social media data, the model maintained >90% accuracy.





AI

CONCLUSION

CONCLUSION & FUTURE WORK

Achievements:

- Built a custom dataset (3,761 samples)
- 451 Real RSS articles
- 753 test samples
- Achieved 96.81% F1-Score with MLP Deep Learning

Future Improvements:

- Integrate FinBERT (Transformer) to improve 'Neutral' class detection.
- Deploy as a real-time REST API for live trading.