AI - FAKE NEWS DETECTION

NATURAL LANGUAGE PROCESSING & MACHINE LEARNING

Dr Subramani

Mentored
Ramisha Rani K
Ramya Dinesh

PROBLEM STATEMENT

- Fake news spreads very fast through social media
- Manual verification takes time and is not accurate
- Fake news looks similar to real news
- People unknowingly trust and share wrong information
- So, we need an **ai-based automatic fake news detection system**

DATASET INFORMATION

- Dataset File: news.csv
- Total Records: 6,335 rows
- Dataset type: supervised text & classification data
- Total Columns: 3
    - title – News Headline
    - text – Full News Content
    - label – FAKE / REAL

DATA PREPROCESSING

- Removed unnecessary column
- Selected useful columns:
  - input → text
  - output → label
- Converted all text into : lowercase letters
- Removed: stop words (is, the, was, etc.)
- Converted text into numbers using vectorization
- This process improves : model speed, accuracy, learning quality
- DATA SPLIT (70% – 30%)
- Total records: 6,335
- Training data: 70% Testing data: 30%

```
# LOAD DATASET
# ============
df = pd.read_csv("news.csv")
dataset = df.drop("Unnamed: 0", axis=1)
X = dataset["text"]
y = dataset["label"]
```

```
CountVectorizer(stop_words='english')
```

```
# TRAIN TEST SPLIT
# ================
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=53
)
```

MODEL CREATION & WHY VECTORIZERS USED

Why Vectorization is Needed:

- machine learning only understands numbers
- so text must be converted into numerical form

Vectorizers Used:

1. Count Vectorizer

- counts how many times each word appears
- simple and fast
- good for basic models

2. TF-IDF Vectorizer

- gives importance to useful words
- removes common word effect
- gives higher accuracy

3. Hashing Vectorizer

- very fast
- uses fixed memory
- good for large data

Models Used:

- Multinomial Naive Bayes
- Passive Aggressive Classifier
- Logistic Regression
- Linear SVM
- Random Forest

- Total 15 model combinations trained
- All models compared using accuracy score

```
#  DEFINE VECTORIZERS
# ===================
vectorizers = {
    "COUNT": CountVectorizer(stop_words="english", max_features=5000),
    "TFIDF": TfidfVectorizer(stop_words="english", max_features=5000),
    "HASHING": HashingVectorizer(stop_words="english", n_features=2**14,
                                 alternate_sign=False)
}

#  DEFINE MODELS
# ===============
models = {
    "MULTINOMIAL NB": MultinomialNB(),
    "PASSIVE AGGRESSIVE": PassiveAggressiveClassifier(max_iter=1000),
    "LOGISTIC REGRESSION": LogisticRegression(max_iter=1000),
    "LINEAR SVM": LinearSVC(),
    "RANDOM FOREST": RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1)
}
```

MODEL VALIDATION, COMPARISON & BEST MODEL SELECTION

- Validation Method:
  - Dataset split into **70% training** and **30% testing**
  - All models evaluated using **accuracy score**
  - Same test data used for **fair comparison**
  - Each vectorizer + model combination tested

```python
#  TRAIN ALL MODELS
# ================================
results = []
for vec_name, vectorizer in vectorizers.items():
    # FIT TRANSFORM (except Hashing)
    if vec_name == "HASHING":
        x_train_vec = vectorizer.transform(x_train)
        x_test_vec = vectorizer.transform(x_test)
    else:
        x_train_vec = vectorizer.fit_transform(x_train)
        x_test_vec = vectorizer.transform(x_test)
    for model_name, model in models.items():
        # Skip invalid combination: Hashing + Multinomial NB
        if vec_name == "HASHING" and model_name == "MULTINOMIAL NB":
            pass
        #  TRAIN MODEL
        model.fit(x_train_vec, y_train)
        y_pred = model.predict(x_test_vec)
        acc = accuracy_score(y_test, y_pred)
        results.append([f"{vec_name} + {model_name}", acc])
```

Final Ranked Accuracy Comparison Table

Rank	Model Combination	Accuracy
1	**TF-IDF + Linear SVM**	**93.37%**
2	TF-IDF + Passive Aggressive	92.42%
3	Hashing + Linear SVM	92.37%
4	Count + Random Forest	91.84%
5	TF-IDF + Random Forest	91.79%
6	Hashing + Random Forest	91.58%
7	Count + Logistic Regression	91.53%
8	Hashing + Passive Aggressive	91.53%
9	TF-IDF + Logistic Regression	91.37%
10	Hashing + Logistic Regression	90.84%
11	Count + Passive Aggressive	89.00%
12	TF-IDF + Multinomial Naive Bayes	88.42%
13	Hashing + Multinomial Naive Bayes	88.32%
14	Count + Linear SVM	88.27%
15	Count + Multinomial Naive Bayes	86.58%

BEST MODEL & FINAL PREDICTION

- **Best Performing Model**
  - TF-IDF + Linear SVM
  - Accuracy: 93.37%
- **Final Prediction Check:**
  - Input: one real news article
  - Actual output: REAL
  - Model predicted: REAL
- prediction is correct and successful

```python
# ACCURACY TABLE WITH RANK
# ==========================
accuracy_df = pd.DataFrame(results, columns=["Model Combination", "Accuracy"])

# Sort by Accuracy (Descending)
accuracy_df = accuracy_df.sort_values(by="Accuracy", ascending=False).reset_index(drop=True)

# Add Rank Column
accuracy_df.insert(0, "Rank", accuracy_df.index + 1)

print("\n =======  FINAL RANKED ACCURACY TABLE =======  \n")
print(accuracy_df.to_string(index=False))   #
```

```python
#  SAMPLE PREDICTION DEMO
# =======================
sample_text = dataset["text"].iloc[2]

best_vectorizer = TfidfVectorizer(stop_words="english", max_features=5000)
best_model = LinearSVC()

x_train_best = best_vectorizer.fit_transform(x_train)
x_test_best = best_vectorizer.transform(x_test)

best_model.fit(x_train_best, y_train)

sample_vector = best_vectorizer.transform([sample_text])
prediction = best_model.predict(sample_vector)[0]

print("\n ======== SAMPLE PREDICTION ======== ")
print("News Text:\n", sample_text)
print("\nActual Label:", y.iloc[2])
print("Predicted Label (TF-IDF + SVM):", prediction)
```

CONCLUSION

- Successfully built an AI-based Fake News Detection System
- Used Natural Language Processing & Machine Learning models
- Compared 15 different model combinations
- TF-IDF + Linear SVM achieved the highest accuracy of 93.37%
- The model accurately classified Real and Fake news
- This system helps reduce misinformation in digital platforms

Thank you Dr. Subramani

======= FINAL RANKED ACCURACY TABLE =======

Rank	Model Combination	Accuracy
1	TFIDF + LINEAR SVM	0.933719
2	TFIDF + PASSIVE AGGRESSIVE	0.924250
3	HASHING + LINEAR SVM	0.923724
4	COUNT + RANDOM FOREST	0.918464
5	TFIDF + RANDOM FOREST	0.917938
6	HASHING + RANDOM FOREST	0.915834
7	COUNT + LOGISTIC REGRESSION	0.915308
8	HASHING + PASSIVE AGGRESSIVE	0.915308
9	TFIDF + LOGISTIC REGRESSION	0.913730
10	HASHING + LOGISTIC REGRESSION	0.908469
11	COUNT + PASSIVE AGGRESSIVE	0.890058
12	TFIDF + MULTINOMIAL NB	0.884271
13	HASHING + MULTINOMIAL NB	0.883219
14	COUNT + LINEAR SVM	0.882693
15	COUNT + MULTINOMIAL NB	0.865860

======== SAMPLE PREDICTION ========

```
Actual Label: REAL
Predicted Label (TF-IDF + SVM): REAL
```