Fake News Detector

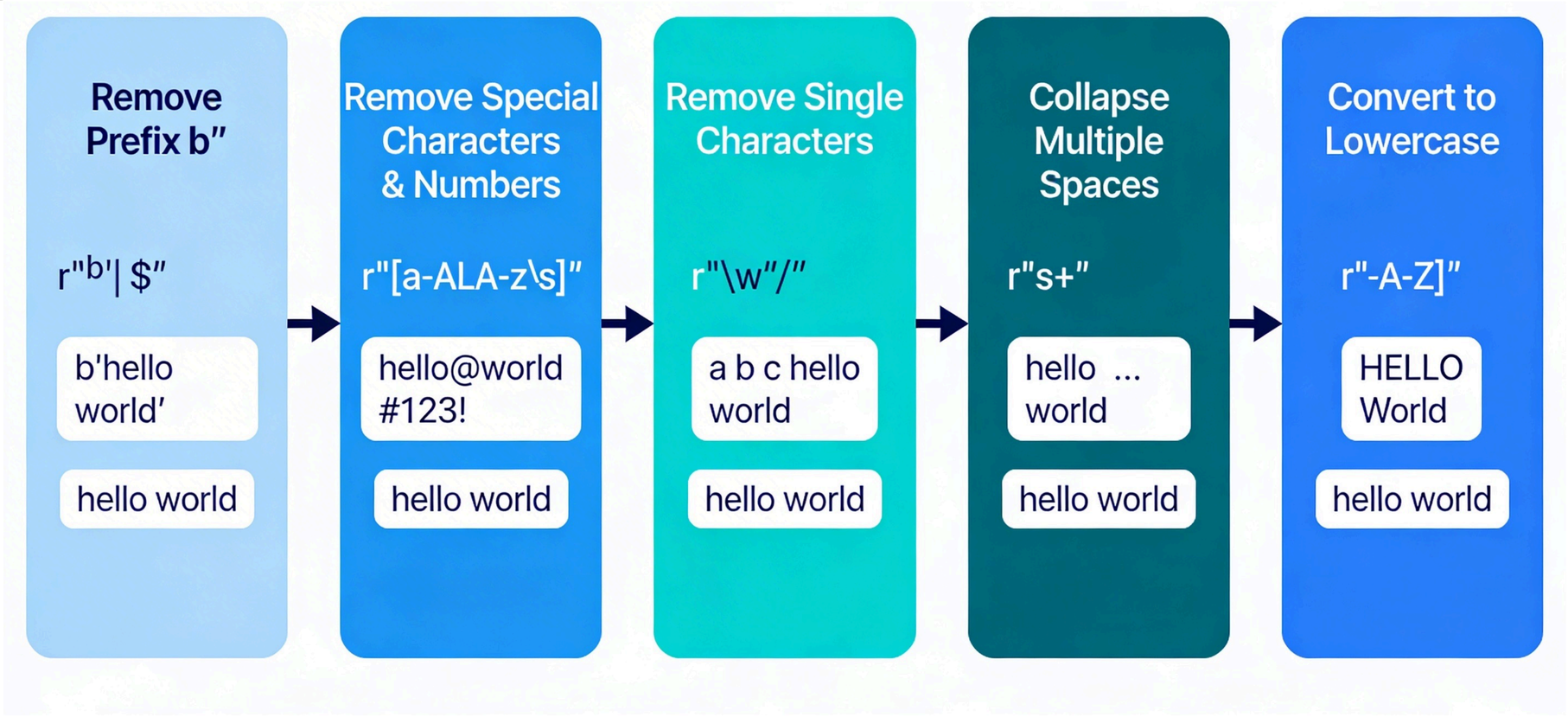
Using NLP techniques

Anne Valvezan
Harmandeep Singh

Ironhack— February 2026

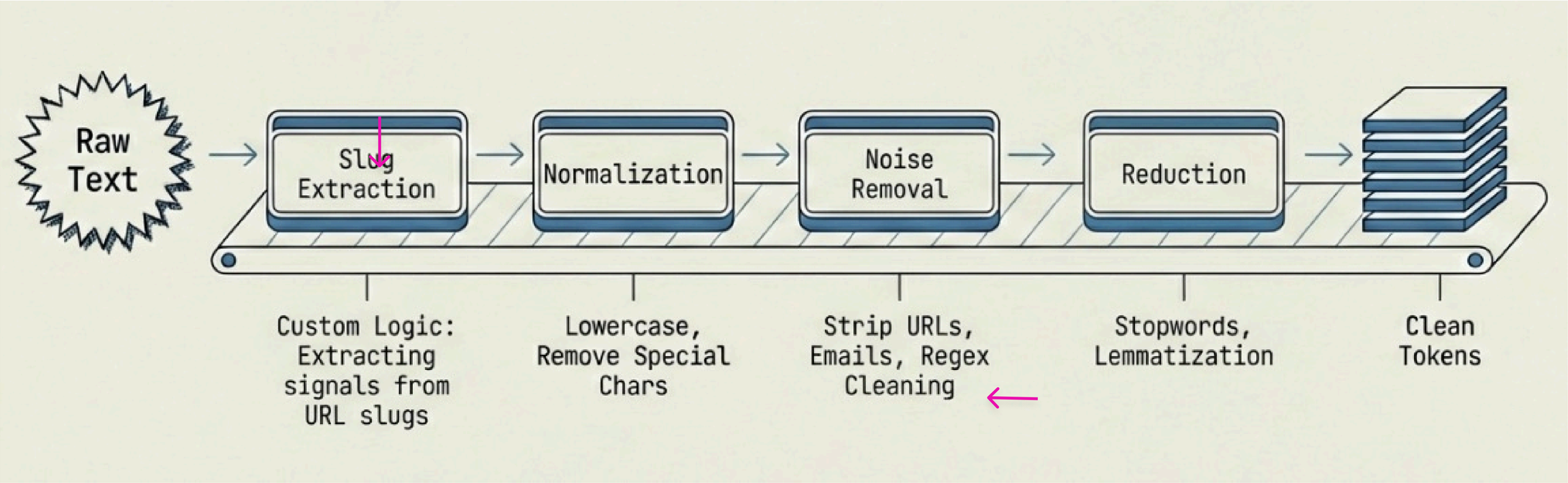
Preprocessing Pipeline

Data Cleaning v1



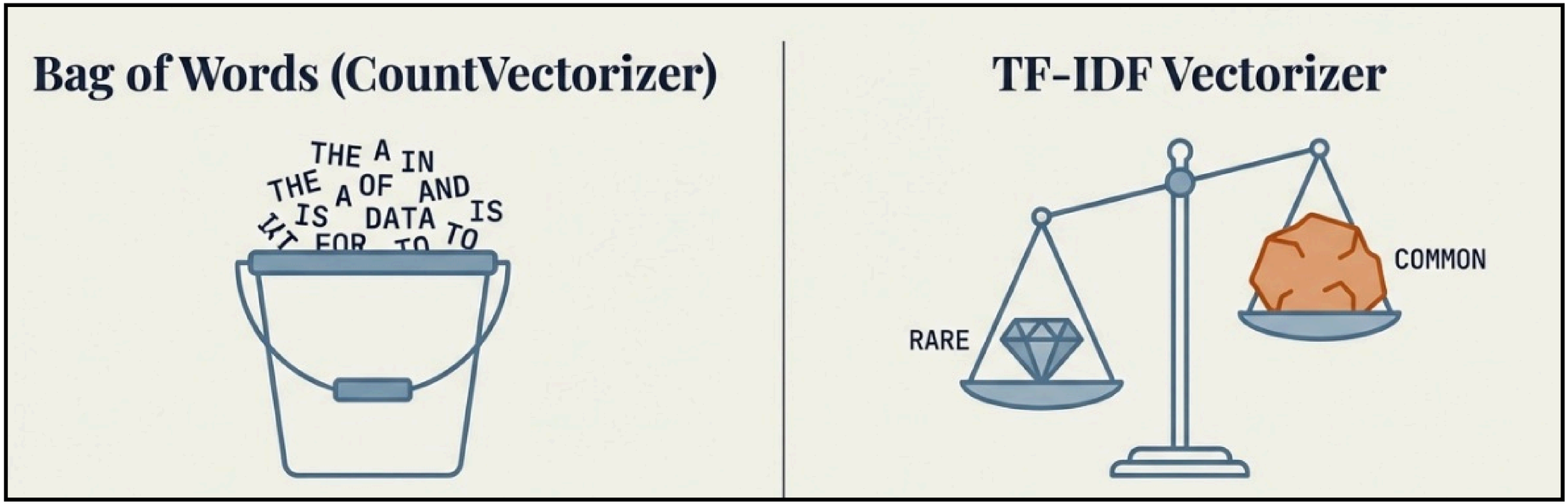
Preprocessing Pipeline

Data Cleaning v2




```
https://100percentfedup.com/served-roy-moore-vietnamletter-veteran-sets-record-straight-honorabl...
https://100percentfedup.com/video-hillary-asked-about-trump-i-just-want-to-eat-some-pie/
https://100percentfedup.com/12-yr-old-black-conservative-whose-video-to-obama-went-viral-do-you-...
```

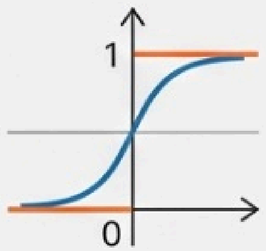
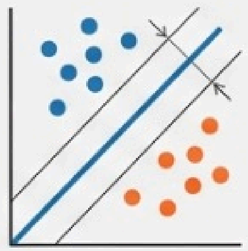
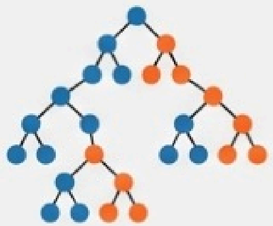
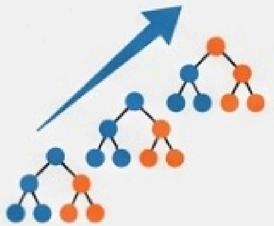
Vectorizers



Frequency of the words appearance in the dataset

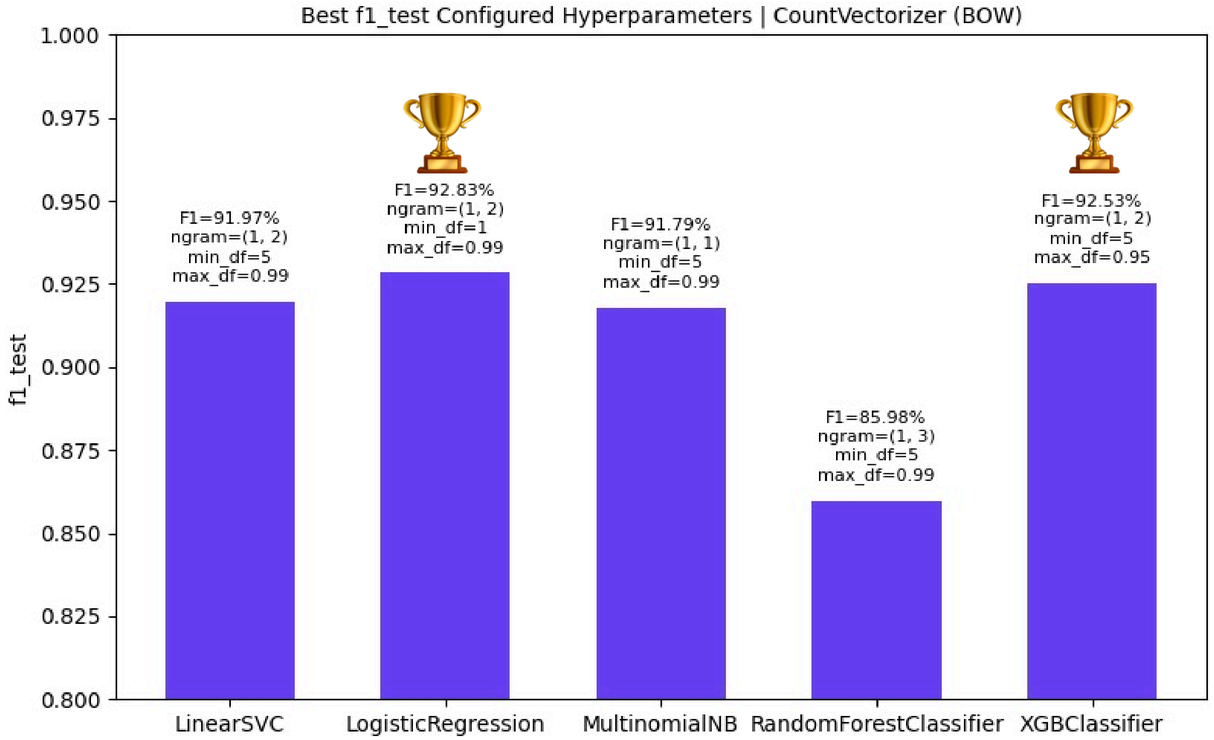
Frequency of the words appearance and applies a factor of rarity in the dataset

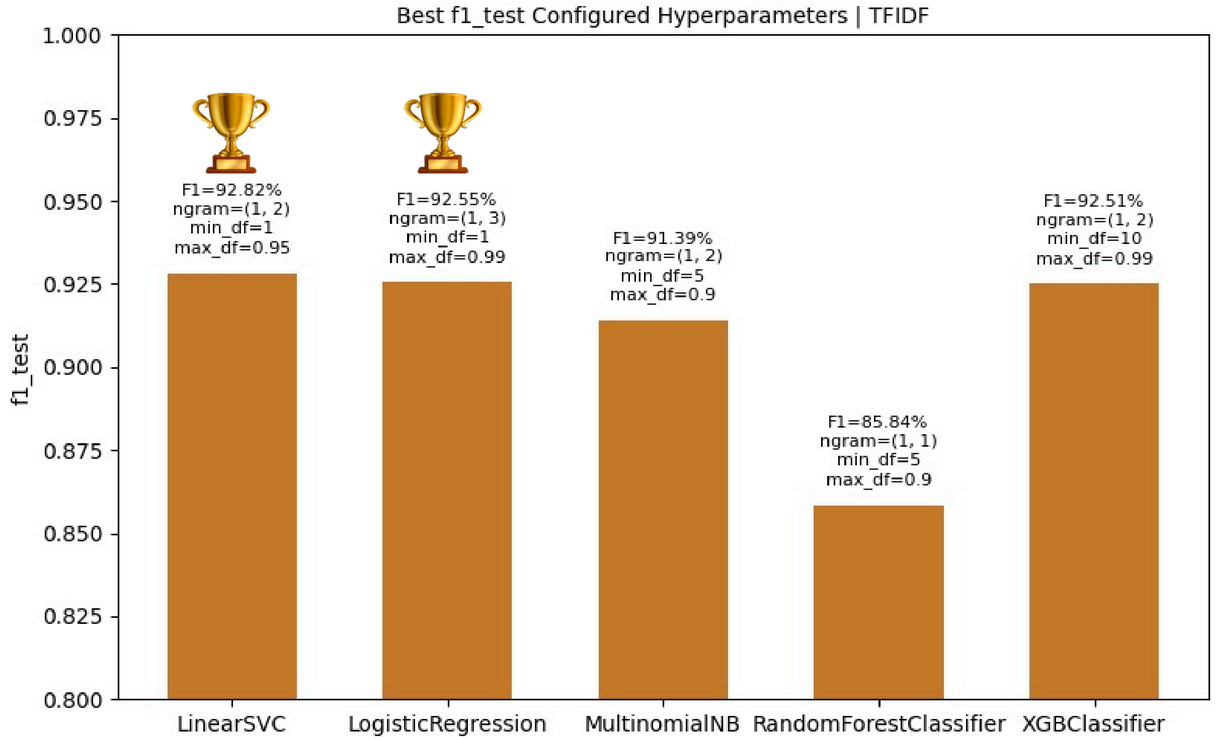
Classifiers

Logistic Regression	LinearSVC	Multinomial NB	Random Forest	XGBoost
				
Linear	Linear	Probabilistic	Ensemble	Ensemble

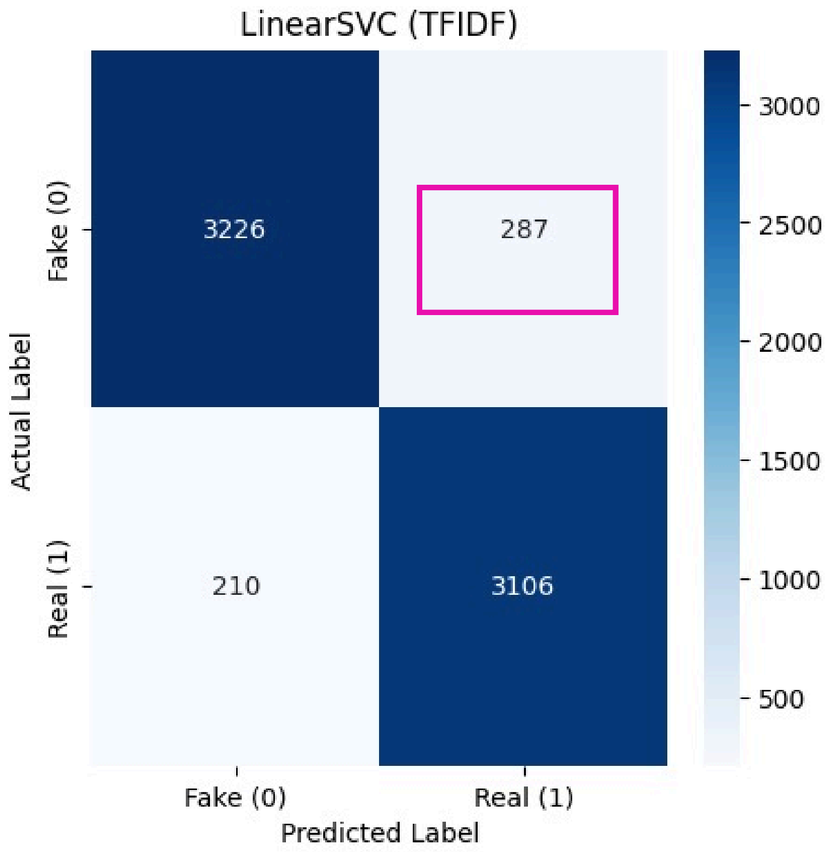
CountVectorizer Vs TFIDF

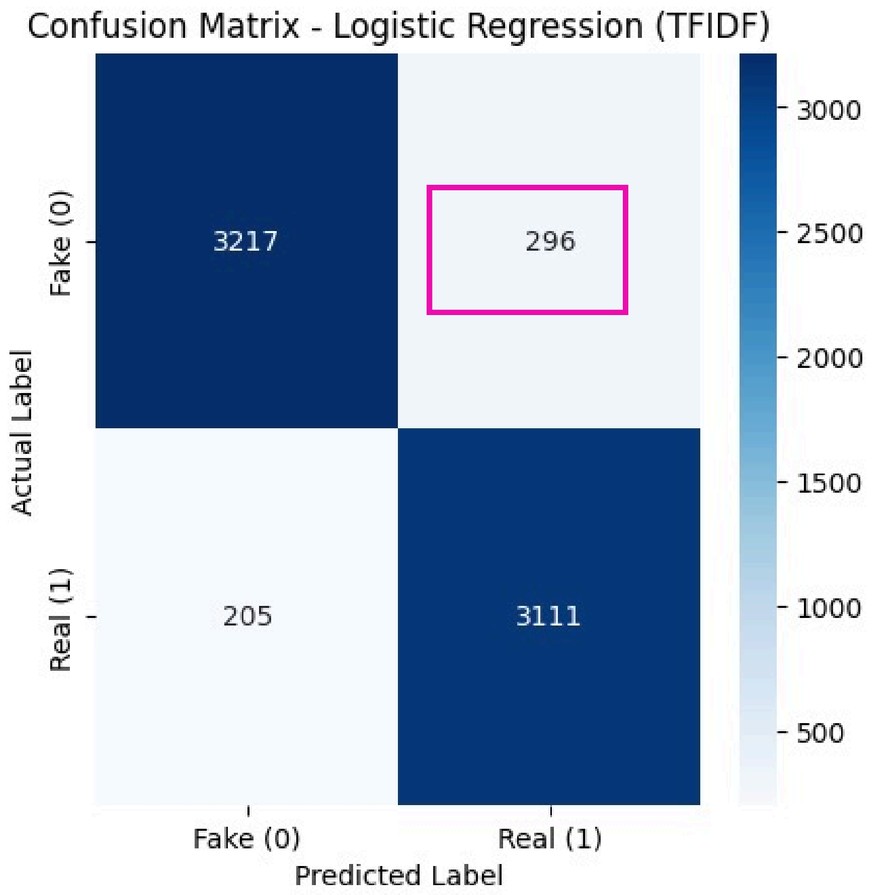
```python
models = [
    LogisticRegression(),
    LinearSVC(),
    MultinomialNB(),
    RandomForestClassifier(n_estimators= 500, max_depth= 10),
    XGBClassifier(n_estimators= 500, max_depth= 10)
]
ngram_vals = [(1, 1), (1, 2), (1, 3), (2, 2)]
min_df_vals = [1, 5, 10, 20]
max_df_vals = [0.90, 0.95, 0.99]
```





TFIDF Best Models Confusion Matrics





Hyperparameter Tunning

RandomizedSearchCV

```python
# ===========Hyperparameter Search=============

SEARCH_SPACES = {
    "random_forest": {
        "classifier": RandomForestClassifier(),
        "param_distributions": {
            "vectorizer": [TfidfVectorizer(), CountVectorizer()],
            "vectorizer__ngram_range": [(1,1), (1,2)],
            "vectorizer__max_features": [1000, 5000],
            "classifier__n_estimators": [100, 200, 400],

            "classifier__criterion": ["gini", "entropy"],
        },
    },
    "logreg": {
        "classifier": LogisticRegression(max_iter=2000),
        "param_distributions": {
            "vectorizer": [TfidfVectorizer(), CountVectorizer()],
            "vectorizer__ngram_range": [(1,1), (1,2)],
            "vectorizer__max_features": [1000, 5000],

            "classifier__C": [0.1, 1.0, 10.0],
        },
    },
    "xgb": {
        "classifier": XGBClassifier(eval_metric="mlogloss"), # cross-entropy loss
        "param_distributions": {
            "vectorizer": [TfidfVectorizer(), CountVectorizer()],
            "vectorizer__ngram_range": [(1,1), (1,2)],
            "vectorizer__max_features": [1000, 5000],

            "classifier__n_estimators": [100, 200, 400],
            "classifier__max_depth": [3, 6, 9],
            "classifier__learning_rate": [0.05, 0.1, 0.3],
        },
    },
    "svn": {
```

```python
print("Starting experiments...")

for model_key, spec in SEARCH_SPACES.items():

    pipe = Pipeline([
        ("vectorizer", TfidfVectorizer()), # placeholder
        ("classifier", spec["classifier"]), ## placeholder
    ])

    # Randomized search
    search = RandomizedSearchCV(
        pipe,
        param_distributions=spec["param_distributions"],
        n_iter=30,
        cv=3,
        scoring="f1_macro",
        n_jobs=-1,
        verbose=1,
        random_state=seed,
    refit=True
    )

    search.fit(X_train, y_train)
```

Hyperparameter Tunning

RandomizedSearchCV

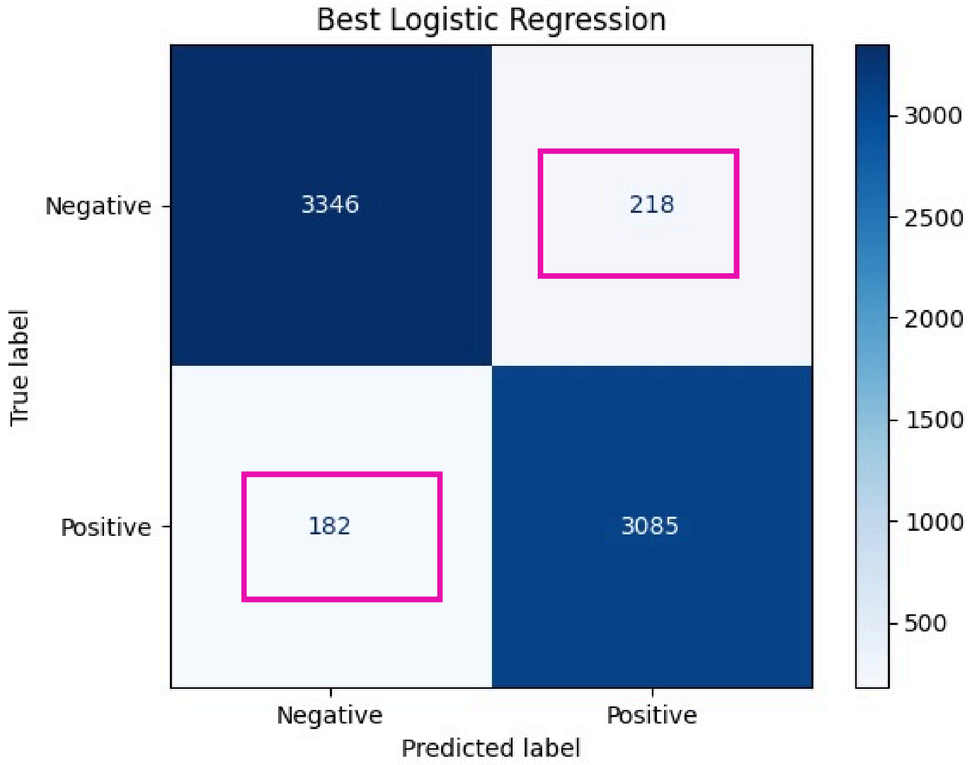
	model	vectorizer	train_accuracy	train_f1_score	test_accuracy	test_f1_score
	RandomForestClassifier	TfidfVectorizer	1.0	1.0	0.9216805738544869	0.9215772080019559
🏆	LogisticRegression	TfidfVectorizer	0.9796493539767944	0.979638698512292	0.9414434197042892	0.9413578100734765
	XGBClassifier	TfidfVectorizer	0.9853592474653197	0.9853543618844387	0.9329527155614112	0.93289690628073
🏆	LinearSVC	TfidfVectorizer	0.9711577175066799	0.9711459816088661	0.9417362026057678	0.9416729225750246
	MultinomialNB	CountVectorizer	0.9489769774166392	0.9489255435984479	0.9310496267018006	0.9308400893055497

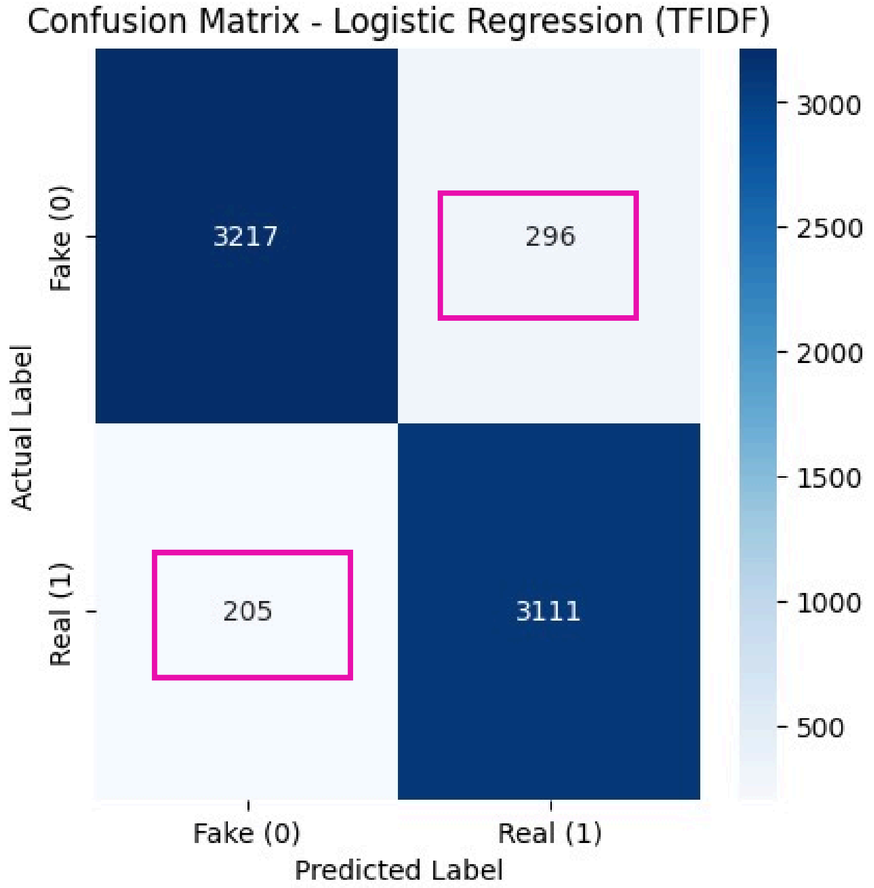
Best parameters:

Logistic Regression : {'vectorizer__ngram_range': (1, 1), 'vectorizer__max_features': 5000, 'classifier__C': 10.0}

Linear SVC : {'vectorizer__ngram_range': (1, 1), 'vectorizer__max_features': 5000, 'classifier__penalty': 'l2', 'classifier__C': 0.3}

TFIDF Best Model Confusion Matrics





Conclusions

- Dataset influenced the results: simpler was better
- LogisticRegression and LinearSVC 🏆
- TF-IDF 🏆
- importance of hyperparameter tunning (increase 1-2%)

Best parameters:

Logistic Regression : {'vectorizer__ngram_range': (1, 1), 'vectorizer__max_features': 5000, 'classifier__C': 10.0}

Linear SVC : {'vectorizer__ngram_range': (1, 1), 'vectorizer__max_features': 5000, 'classifier__penalty': 'l2', 'classifier__C': 0.3}

Thank You

Anne Valvezan
Harmandeep Singh