Real or Fake?

Classifying News Headlines with Natural Language Processing

Laura Salakari - 17.01.2026

Table of Contents

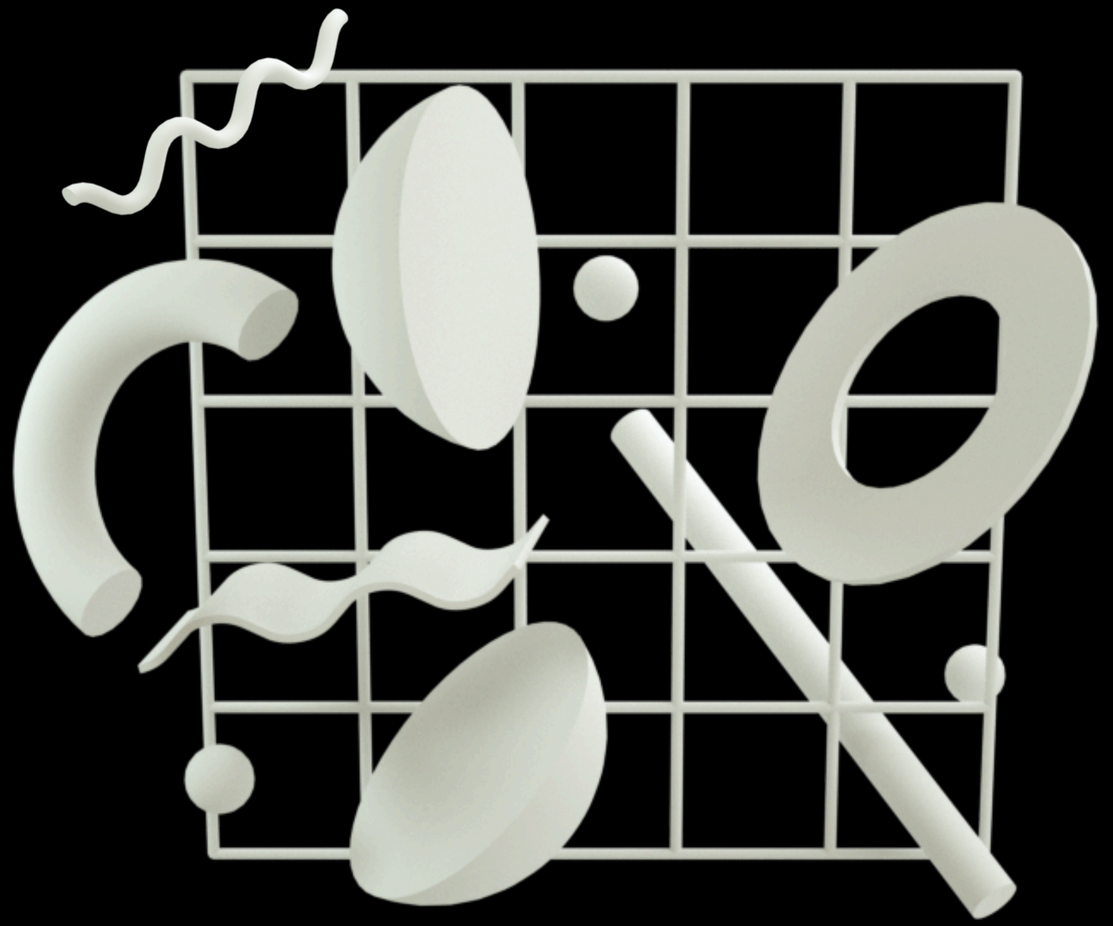
The Data

Training Data

- 34152 entries
- 17572 fake headlines (51.5%)
- 16580 real headlines (48.5%)

Testing Data

- 9984 unlabelled entries



Preprocessing the Data

Preparing the Data for NLP

1. Removing punctuation
2. Lowercasing the headlines
3. Removing stopwords
4. Splitting the training data (80/20)
5. Lemmatization

stopwords.py

```python
stop_words = set(stopwords.words('english'))

def remove_stop_words(text):
    return " ".join(
        w for w in text.lower().split()
        if w not in stop_words
    )

df_train["headline"] = df_train["headline"].apply(remove_stop_words)
```

Stopword code

lemmatization.py

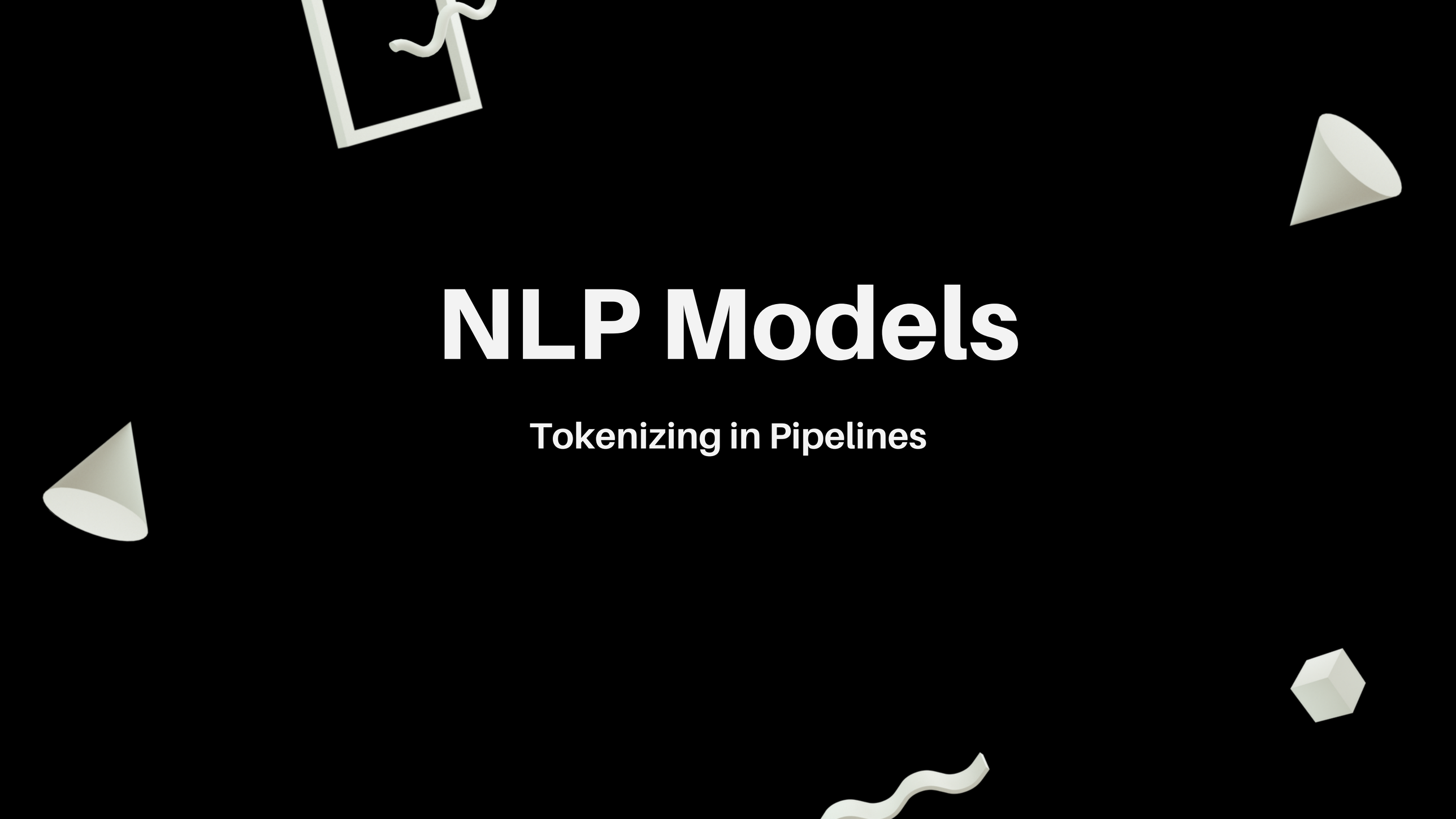
```python
import spacy

nlp = spacy.load("en_core_web_sm", disable=["ner", "parser"])

def lemmatize(text):
    if not isinstance(text, str):
        return ""
    doc = nlp(text)
    return " ".join(token.lemma_ for token in doc if token.is_alpha)

X_train = X_train.apply(lemmatize)
```

Lemmatization code


NLP Models
Tokenizing in Pipelines

Common steps

All models were built with Scikit’s Pipeline system and include GridSearches to improve model performance.
All Pipelines begin with Tf-Idf tokenization and vectorization

Random Forest Classifier

rfc.py

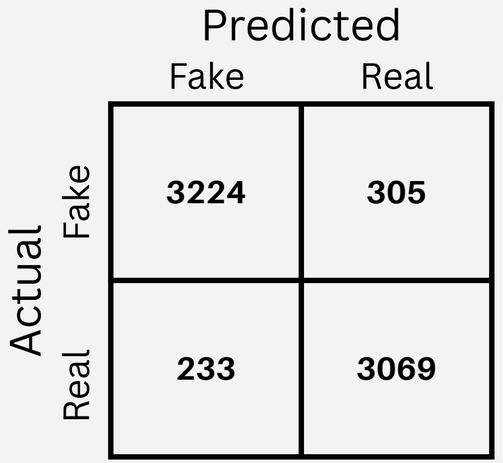
```python
rf_pipeline = Pipeline([
    ("tfidf", TfidfVectorizer()),
    ("rfc", RandomForestClassifier(n_estimators=200, criterion="entropy", random_state=42))
])

rf_param_grid = {
    "tfidf__ngram_range": [(1,1), (1,2), (1,3)],
    "tfidf__min_df": [1,2,5],
    "tfidf__max_df": [0.85, 0.9, 0.95],
}

rf_grid = GridSearchCV(rf_pipeline, rf_param_grid, scoring="accuracy", cv=5, verbose=2)
rf_grid.fit(X_train, y_train)
```

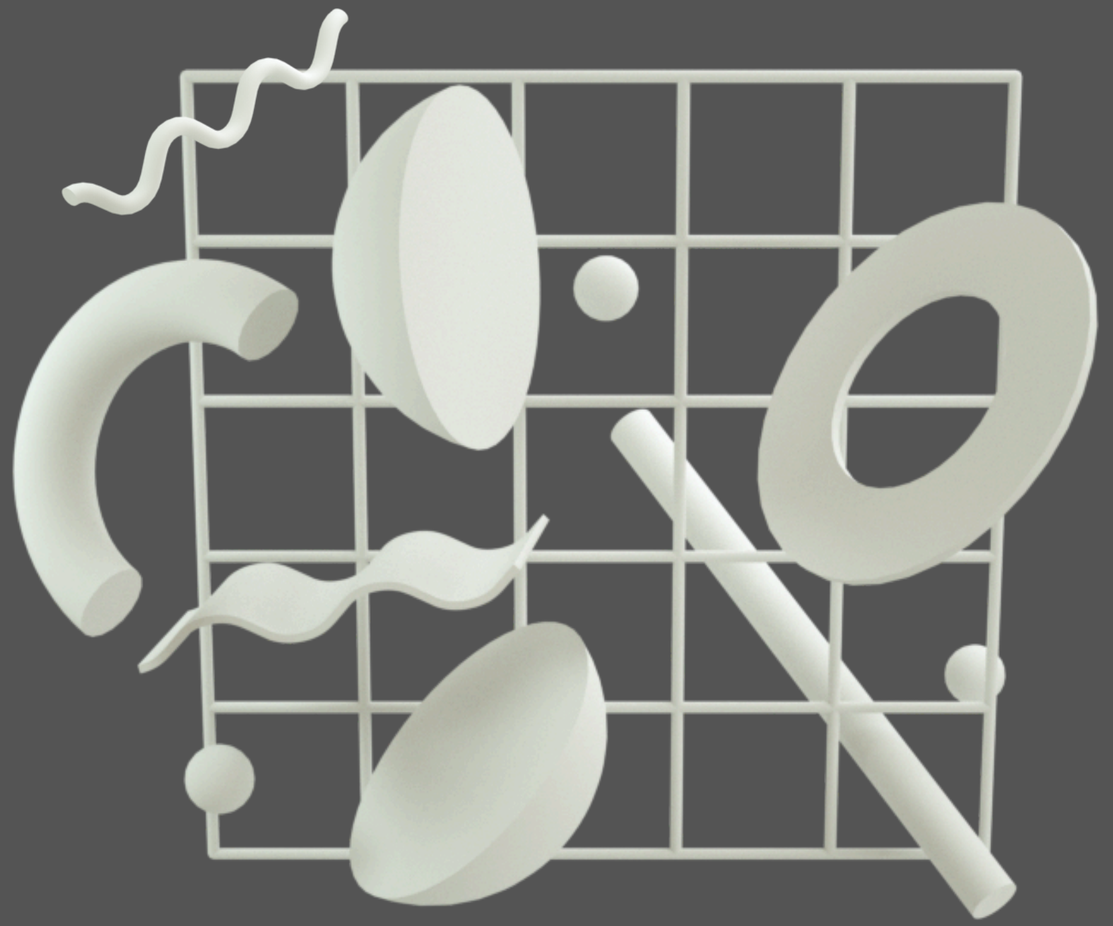
1. Tf-Idf with grid search to find best tokenization and vectorization
2. Fitting RandomForest with training data

Predictions on the model with the best estimators from the pipeline yield an accuracy of **0.9212**



Tf-Idf Params

- ngram_range = (1, 3)
- min_df = 2
- max_df = 0.85



Logistic Regression

lr.py

```python
lr_pipeline = Pipeline([
    ("tfidf", TfidfVectorizer(max_df=0.85, min_df=2, ngram_range=(1, 3))),
    ("lr", LogisticRegression(random_state=42, max_iter=500))
])

lr_params = {
    "lr__C": [0.1, 1, 10],
}

lr_grid = GridSearchCV(lr_pipeline, lr_params, scoring="accuracy", cv=5, verbose=2)
lr_grid.fit(X_train, y_train)
```

1. Tf-Idf with best params from RFC to find best tokenization and vectorization
2. Grid Search for best C value in Logistic Regression (10)
3. Fitting with best params

Predictions on the model with the best estimators from the pipeline yield an accuracy of **0.9413**

		Predicted: Fake	Predicted: Real
Actual	Fake	**3311**	**218**
	Real	**183**	**3119**

LinearSVC

lsvc.py

```python
lsvc_pipeline = Pipeline([
    ("tfidf", TfidfVectorizer(max_df=0.85, min_df=2, ngram_range=(1, 3))),
    ("lsvc", LinearSVC(random_state=42, class_weight="balanced"))
])

lsvc_grid_params = {
    "lsvc__C": [0.01, 0.1, 1, 10]
}

lsvc_grid = GridSearchCV(lsvc_pipeline, lsvc_grid_params, scoring="accuracy", cv=5, verbose=2)
lsvc_grid.fit(X_train, y_train)
```

1. Tf-Idf with best params from RFC to find best tokenization and vectorization
2. Grid Search for best C value in LinearSVC (1)
3. Fitting with best params

Predictions on the model with the best estimators from the pipeline yield an accuracy of **0.9417**

		Predicted: Fake	Predicted: Real
Actual	Fake	**3308**	**221**
	Real	**177**	**3125**

Comparison

Model	Accuracy
LinearSVC	0.9417
Logistic Regression	0.9413
Random Forest	0.9212

* LinearSVC fared slightly better with classifying real headlines correctly, while Logistic Regression had an edge with fake headline

Tackling the Testing Data

Preparing for classification

Before proceeding with classification, the testing data needs to be preprocessed the same way as the training data was:

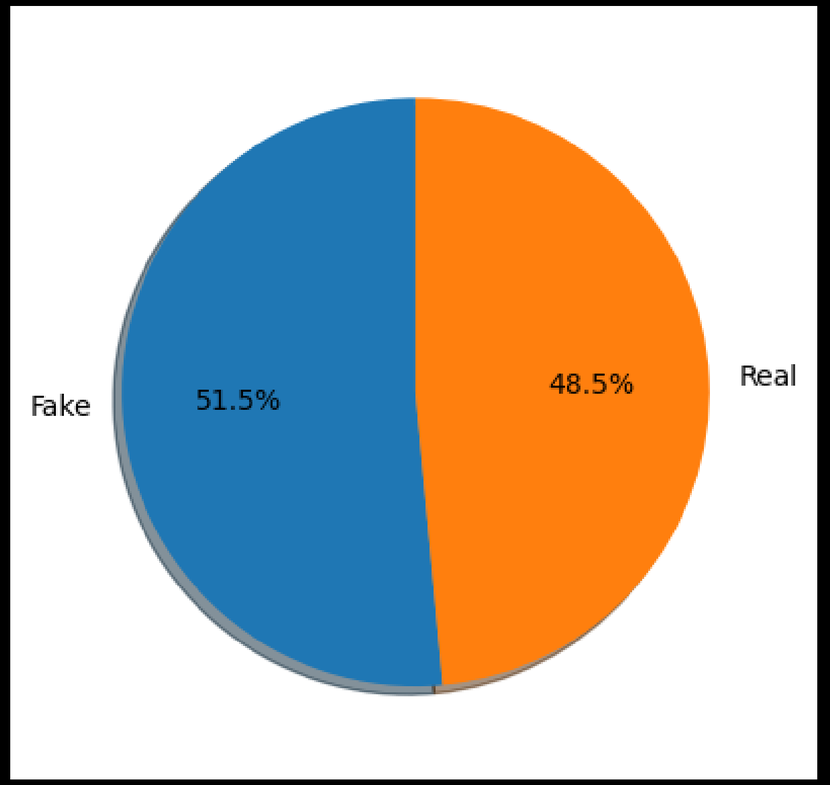
1. Punctuation
2. Lowercasing
3. Stopwords
4. Lemmatization

Classification is conducted with the best model from LinearSVC as it had the highest accuracy

testing_data.csv is overwritten with the new labels accompanying the original , unprocessed headlines

Testing Results

51.5% of the headlines (5138) are classified as fake, leaving 48.5% (4846) as real headlines.



Thank you for your attention