Project on:

"Identification of Fake Product Reviews"

By

Rome Delmo

Table of Content

Abstract

Detecting fake online reviews in e-commerce is even more challenging because of increased review sophistication and volume. This project explores AI for accurate and automated identification of fake reviews. NLP techniques extracted linguistic, behavioural, and sentiment features from reviews. Using 44,000 labelled reviews, traditional supervised machine learning approach was adopted (LR, NB, DT, SVM, RF, AdaBoost, KNN, and Stacking). AdaBoost was identified as the optimal model; when hyperparameter-tuned, it achieved 82% accuracy and 91% ROC AUC on the test set. This means the model effectively distinguished between real and fake reviews. In conclusion, the project achieved all its outlined objectives. Future work includes using more powerful computational resources, exploring advanced features like n-grams and user metadata, and comparing performance with transformer models.

List of Abbreviations

AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
AUC	Area Under the Curve
CPU	Central Processing Unit
DT	Decision Tree
GPU	Graphics Processing Unit
KNN	K-nearest Neighbours
LR	Logistic Regression
ML	Machine Learning
NB	Naïve Bayes
NLP	Natural Language Processing
POS	Part-of-Speech
RAM	Random Access Memory
RF	Random Forest
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine

1. Introduction

One growing problem in the domain of e-commerce is fake reviews, they threaten consumer trust and product reliability. Manual human detection is insufficient because of increasing sophistication and volume of online reviews. This project intends to address these challenges using AI for accurate and automated identification.

2. The Real-World Problem

Fake online reviews are a significant real-world problem. A UK Regulatory body called the Competition and Markets Authority (CMA) has launched investigations into platforms like Google and Amazon for not doing enough to protect consumers from misleading reviews (Competition and Markets Authority 2020). Economically, fake reviews create false claims of product quality, leading to higher return rates and increased operational costs like shipping, inventory, storage, packaging, and labour. Environmentally, excessive product returns increase transportation emissions and packaging waste. Ethically, deceptive reviews can cause financial harm or pose health and safety risks, particularly for sensitive products like health supplements.

3. Project Aim and Objectives

It is difficult for humans to distinguish fake reviews from genuine reviews; hence, the need for assistance using AI; this project aims to deploy intelligent systems to accurately identify fake reviews.

Objectives

1. Collect credible datasets from the internet ***(Achieved in 4.1.1)***
2. Preprocess data: Clean data and extract relevant features using NLP ***(Achieved in 4.1.3.1)***
3. Model selection: Use cross-validation to evaluate different models and select one with the best performance ***(Achieved in 4.2.1)***
4. Hyperparameter tuning: Use Random Search to find the best parameters ***(Achieved in 4.2.2.1)***
5. Model Training and Evaluation: Evaluate performance of final model on unseen test set ***(Achieved in 4.2.2.2)***

4. Adopted Artificial Intelligence Approach

This project has adopted a traditional supervised ML approach to identify fake reviews. This is the best choice because the dataset used are labelled data for effective model training. Unsupervised learning is less suitable for this problem because it is more challenging to validate if reviews are fake using unlabelled dataset.

Traditional models such as LR or DT were chosen for interpretability; this means it is easier to see how the models arrived at the conclusion that a review is fake or not. The disadvantage of using deep learning is their lack of interpretability, this means that the multiple layers of transformations in neural networks make it difficult to understand how specific features, like review content or length, contribute to the classification of fake reviews.

Literature	**Dataset**	**Accuracy (%)**						
		LR	**NB**	**DT**	**RF**	**SVM**	**KNN**	**AdaBoost**
Abdulqader *et al.* (2022)	YelpZip (608,598 instances)	86.78	75.74	80.60	86.37	-	-	-
Alsubari *et al.* (2020)	Yelp electronic product reviews (30,476 instances)	-	-	**96**	94	-	-	**97**
Alsubari *et al.* (2022)	Hotel reviews from Trip Advisor (1,600 instances)	86	**88**	-	**95**	**93**	-	94
Elmogy *et al.* (2021)	Hotel reviews from Yelp (5,853 instances)	86.89	86.08	-	86.82	86.9	**86.23**	-
Jain *et al.* (2021)	Hotel reviews from Yelp (number of instances not available)	**88.11**	80.71	84.57	-	74.55	82.51	85.68
Shan *et al.* (2021)	Restaurant reviews from Yelp (24,539 instances)	-	73.5	-	92.9	84.9	-	-

Figure 1 Literature review with achieved accuracies

To further support my choice, credible literatures support the effectiveness of traditional supervised ML approach for identifying fake reviews, summarised in Figure 1. The best performance achieved was using AdaBoost with 97% accuracy by Alsubari *et al.* (2020). Furthermore, Abdulqader *et al.* (2022) demonstrated that even with large datasets of 608,598 instances, traditional models can still achieve impressive accuracy.

4.1 Artificial Intelligence Approach Implementation

4.1.1 Problem definition

Fake review identification is a binary classification task because reviews are labelled as real or fake. Figure 2 shows the dataset from Salminen (2025) that has labels CG (computer-generated) and OR (original-review), making it a supervised ML problem requiring NLP to extract features from text. A word cloud in Figure 3 provides insight into word patterns. Figure 4 confirms a balanced dataset: this is beneficial because there is a reduced bias towards a majority class.

```
df.tail() # inspection
```

	category	rating	label	text_
40427	Clothing_Shoes_and_Jewelry_5	4.0	OR	I had read some reviews saying that this bra r...
40428	Clothing_Shoes_and_Jewelry_5	5.0	CG	I wasn't sure exactly what it would be. It is ...
40429	Clothing_Shoes_and_Jewelry_5	2.0	OR	You can wear the hood by itself, wear it with ...
40430	Clothing_Shoes_and_Jewelry_5	1.0	CG	I liked nothing about this dress. The only rea...
40431	Clothing_Shoes_and_Jewelry_5	5.0	OR	I work in the wedding industry and have to wor...

Figure 2 Last five instances of dataset

Figure 3 Word cloud

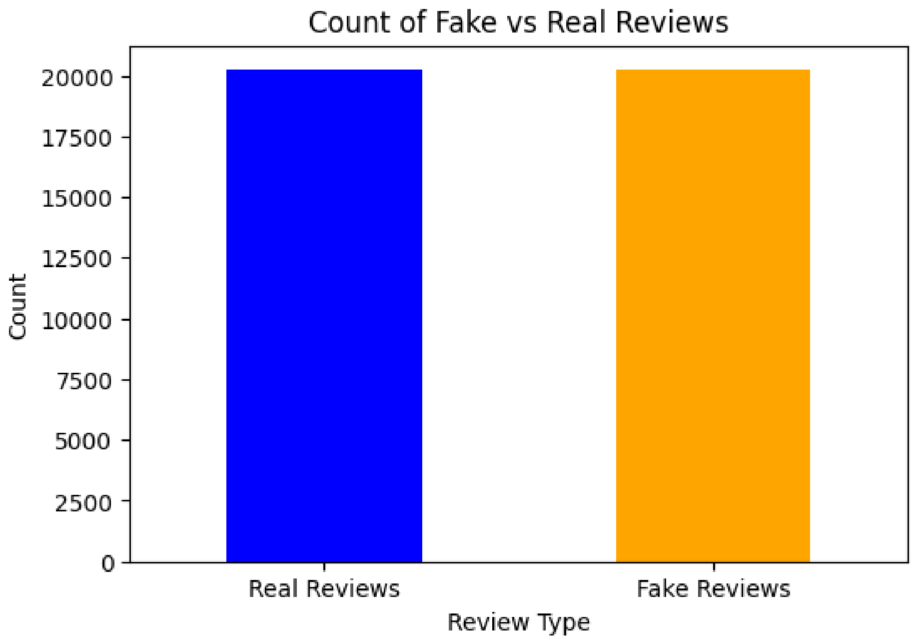

Figure 4 Review Distribution (20,216 real and 20,216 fake)

Natural Language Toolkit (NLTK) and SpaCy are used for NLP; whereas, Scikit-learn is used for ML, shown in Figure 5.

```
import re
import joblib
import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

# Natural Language Processing (NLP)
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.sentiment import SentimentIntensityAnalyzer
from collections import Counter
from wordcloud import WordCloud
import spacy

# Downloading required NLTK datasets
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('universal_tagset')
nltk.download('vader_lexicon')

# Machine Learning and Preprocessing
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split, cross_val_score, cross_validate, RandomizedSearchCV
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, roc_auc_score
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, StackingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline

```

Figure 5 Imported libraries

4.1.2 Hypothesis and AI Approach

The hypothesis is that supervised ML models trained on extracted linguistic, behavioural, and sentiment features will accurately classify real and fake reviews. Figure 6 illustrates workflow closely following project objectives. The approach by K P *et al.* (2024) informs the choice of extracted features.

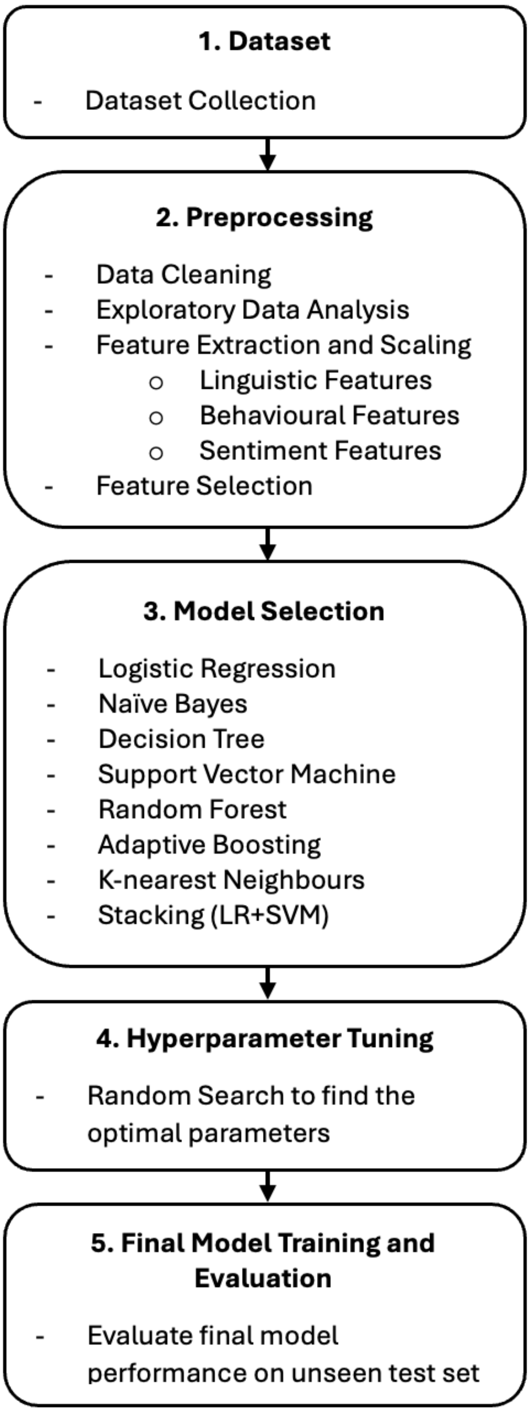

Figure 6 Flow diagram of AI Approach

Classifiers used in our AI approach to identify fake reviews and explanation their selection are outline in Figure 7. LR, NB, and DT serve as baselines. Linear, probabilistic, tree-based, margin-based, and instance-based models are included. Ensemble methods like stacking, boosting, and voting are also employed, providing diversity for model selection.

Classifier	Approach	Justification for Selection
LR	Linear model using weighted sum of input features plus a bias term, a sigmoid function is applied to produce probabilities of how likely reviews are fake. Reviews are classified fake if probability exceeds a set threshold (p=0.5).	Baseline model, fast training, good for high-dimensional text data, and effective in binary classification.
NB	Probabilistic model using Bayes' Theorem to calculate probabilities of input features, like word frequency or sentiment scores, to estimate if a review is fake. A review is assigned two probabilities, one for being fake and one for real, and is decided based on the higher probability.	Baseline model, fast, handles noisy text through feature independence, efficient with high-dimensional data, effective in binary classification.
DT	Tree-based model that splits data at nodes based on input features for creating smaller groups that are progressively better at classifying reviews as fake or real. The classification is decided by following the tree's path until reaching the leaf node which labels the review as fake or real.	Baseline model, fast, interpretable as you can visualise decision tree structure, can identify feature importance, suitable for classification.
SVM	Margin-based model which maps input features into a high-dimensional space to find the optimal hyperplane, a decision boundary which best separates fake and real reviews. The decision is made based on which side of the hyperplane a review falls on, classifying whether it is fake or real.	Effective for large number of features like this study, can handle non-linear data, generalises well, resilient to overfitting.
RF	Voting ensemble model which combines multiple DT, each tree is trained on a random subset of the dataset and features. RF aggregates the predictions from all trees and uses majority voting to decide if a review is fake or real.	Due to ensemble learning, it is expected to have improved accuracy, and more resilient to overfitting and outliers than DT.
AdaBoost	Boosting ensemble model trains weak learners sequentially, each model corrects the errors made by the previous. Classification is made by combining the weighted votes from all models, more accurate models has a greater effect on deciding if a review is fake or real.	Expected to have improved accuracy, has inherent feature selection. Importance of misclassified reviews is adjusted during training to focus on difficult instances in each iteration.
KNN	Instance-based model use Euclidean distance between a target review and training reviews based on input features to identify the K closest neighbours, where K is the number of nearest training reviews considered. The review is classified as fake or real based on the most common label of these neighbours.	Simple, can be used for classification tasks, more flexible for text patterns because it is non-parametric, combined localised boundary formation produce highly adaptable global decision boundary.
Stacking	Stacking ensemble classifier in this study implemented SVM and LR as base models, their predictions are used as input for a LR estimator. The decision is made by the final LR meta-classifier based on the combined predictions of the base models.	Reduce individual model weaknesses, combines unique strengths of SVM and LR, reduces overfitting as meta-learner learns from validation predictions and not training set.

Figure 7 Justification of AI Approach

4.1.3 Model Deployment and Performance Validation Techniques

4.1.3.1 Preprocessing

First, the data was cleaned. The dataset did not have any missing values. The irrelevant feature 'Category' was dropped because fake reviews can exist across all categories. Feature encoding was performed on the target variable, shown in Figure 8.

```
# Encoding 'label': CG to 1, OR to 0
df_fe["label"] = df_fe["label"].map({"CG": 1, "OR": 0})
```

Figure 8 Feature Encoding

In Figure 9, the text column was processed by ensuring all characters are strings and lowercase. Unnecessary numbers, punctuations, and symbols are removed.

```
df_fe['text_'] = df_fe['text_'].fillna('') # filling NaN with empty string
df_fe['text_'] = df_fe['text_'].astype(str) # ensuring values are all strings
df_fe['text_'] = df_fe['text_'].str.lower() # lowercase
df_fe['text_'] = df_fe['text_'].str.replace(r'\d+', '', regex=True) # removing numbers
df_fe['text_'] = df_fe['text_'].str.replace(r'[^\w\s]', '', regex=True) # removing punctuations
```

Figure 9 Text Processing

In Figure 10, text is split into individual words, called tokens, to support model training by converting text into structured data that can be analysed for patterns.

```
# Tokenizing text to store in a new column
df_fe['tokenized'] = df_fe['text_'].dropna().apply(word_tokenize)
```

```
df_fe.tail(3)
```

	rating	label	text_	tokenized
40429	2.0	0	you can wear the hood by itself wear it with t...	[you, can, wear, the, hood, by, itself, wear, ...
40430	1.0	1	i liked nothing about this dress the only reas...	[i, liked, nothing, about, this, dress, the, o...
40431	5.0	0	i work in the wedding industry and have to wor...	[i, work, in, the, wedding, industry, and, hav...

Figure 10 Tokenisation

Afterwards, feature extraction was performed. Firstly, two linguistic features are extracted to extract patterns from words and sentences. POS tagging in Figure 11 outputs how often each part of speech (like nouns, verbs, adjectives) appears in the reviews. For instance, a count of 3 for "NOUN" means three nouns are in the review.

```
# Applying POS tagging and counting the types
pt_df = pd.DataFrame([Counter(tag for i, tag in pos_tag(text, tagset='universal')) for text in df_fe['tokenized']]).fillna(0)
```

```
pt_df.head()
```

	VERB	DET	NOUN	CONJ	ADV	ADJ	PRON	ADP	PRT	NUM	X
0	3.0	1.0	5.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
1	2.0	3.0	5.0	0.0	0.0	2.0	1.0	3.0	0.0	0.0	0.0
2	2.0	3.0	6.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0
3	3.0	2.0	4.0	1.0	1.0	1.0	2.0	2.0	1.0	0.0	0.0
4	5.0	1.0	3.0	1.0	3.0	2.0	1.0	1.0	0.0	1.0	0.0

Figure 11 POS Tagging

The other is syntactic structures shown in Figure 12. This refers to relationships of words in a sentence, and dependency parsing is used to count these relationships. In "The rabbit hop", the counts would be 1 for "nsubj" (subject: rabbit) and 1 for "root" (main verb: hop).

```
nlp = spacy.load("en_core_web_sm") # loading model for dependency parsing
dep_df = pd.DataFrame([Counter(token.dep_ for token in nlp(" ".join(text))) for text in df_fe['tokenized']]).fillna(0)
```

```
dep_df
```

	advcl	nsubj	advmod	relcl	oprd	cc	conj	ROOT	dobj	det	...	predet	appos	prt	expl	preconj	quantmod	csubjpass	agent	case	meta
0	1.0	2.0	3.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	3.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	2.0	0.0	0.0	0.0	1.0	1.0	1.0	2.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 12 Dependency Parsing

Behavioural features were extracted, which include review length and normalising the 'rating' feature. Sentiment features in Figure 13 describe emotional tone of sentences, quantified using negative, neutral, and positive scores. For instance, a strong positive sentiment like "I love using this product!" could have scores neg:0.0, neu:0.2, and pos:0.8.

```
sia = SentimentIntensityAnalyzer()
full_sentiment = df_fe['text_'].apply(sia.polarity_scores) # applying sentiment analysis
full_sentiment_df = pd.DataFrame(full_sentiment.tolist()) # converting list of dictionaries into a DataFrame
```

```
negative_values = (full_sentiment_df < 0).sum() # checking for negative values using boolean
print(negative_values[negative_values > 0])
```

```
compound    4550
dtype: int64
```

```
sentiment_df = full_sentiment_df[['neg', 'neu', 'pos']] # dropping 'compound' column
```

```
sentiment_df.head()
```

	neg	neu	pos
0	0.000	0.258	0.742
1	0.000	0.509	0.491
2	0.000	0.611	0.389

Figure 13 Sentiment analysis

All features are combined. RF was used for plotting each feature's contribution to model prediction in Figure 14. This helps decide which features to remove when improving model performance.

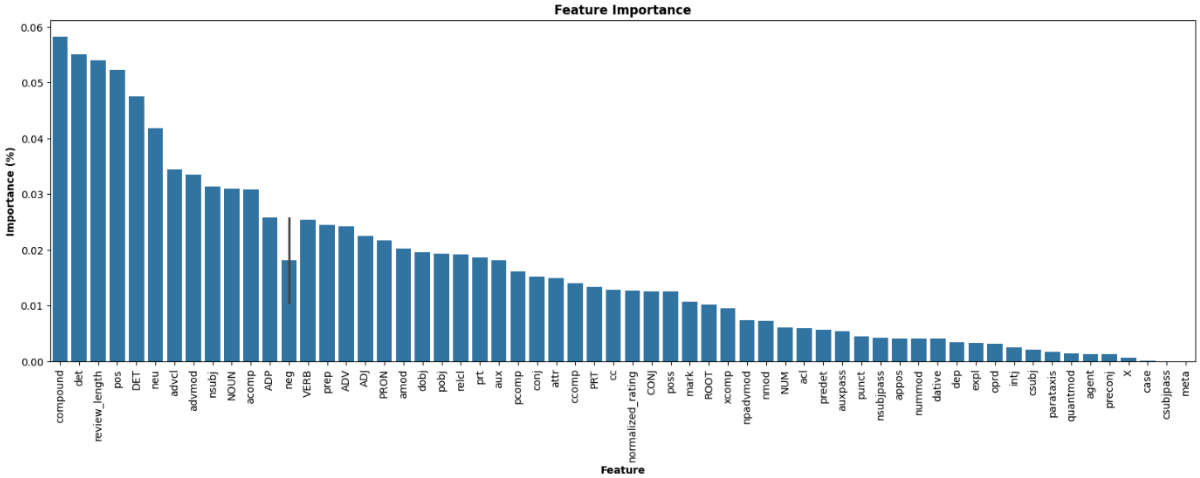

Figure 14 Feature Importance

4.1.3.2 Model Selection

The dataset is split into 80% training and 20% test set. In Figure 15, user-defined function was created for evaluating a single model using 5-fold cross-validation and prints desired performance metrics: accuracy, f1-score, precision, recall, roc_auc, and mean fit time.

```python
def cross_validation(model, X, y, cv=5, model_name="Model"):
    metrics = ['accuracy', 'f1', 'precision', 'recall', 'roc_auc']

    # Cross-validation
    results = cross_validate(model, X, y, cv=cv, scoring=metrics, return_train_score=True, n_jobs=-1)

    # DataFrame for results
    df_results = pd.DataFrame({
        "Metric": metrics,
        "Train Score": [results['train_' + m].mean() for m in metrics],
        "Validation Score": [results['test_' + m].mean() for m in metrics]})
    print(model_name, " Cross-Validation Results:")
    print(df_results.to_string(index=False))
    print("Mean Fit Time:", results['fit_time'].mean())
```

Figure 15 Cross-validation function for programming efficiency

Figure 16 shows an example using LR. A pipeline scaled features between 0 and 1, and initialises the LR classifier. Pipelines prevent data leakage and ensure accurate model evaluation by processing each cross-validation fold independently. Cross-validation of other models followed the same code as Figure 16; results are discussed in Section 4.2.

```python
# Pipeline for scaling and LogisticRegression classifier
pipeline_LR = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('LR', LogisticRegression(random_state=42, solver='saga'))]) # 'saga', good for large datasets

cross_validation(pipeline_LR, X_train, y_train, cv=5, model_name="Logistic Regression")
```

```
Logistic Regression  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.778868          0.777091
       f1     0.782788          0.781203
precision     0.770424          0.768434
   recall     0.795556          0.794444
  roc_auc     0.870728          0.869786
Mean Fit Time: 2.5173527240753173
```

Figure 16 LR Pipeline and cross-validation

4.2 Evaluation, Results and Discussions

4.2.1 Model selection

Model	Accuracy		F1		Precision		Recall		ROC AUC		Time (s)
	T	V	T	V	T	V	T	V	T	V	
LR	0.78	0.78	0.78	0.78	0.77	0.77	0.80	0.79	0.87	0.87	2.5
NB	0.62	0.62	0.68	0.68	0.59	0.59	0.79	0.79	0.68	0.67	0.053
DT	1.00	0.71	1.00	0.71	1.00	0.71	1.00	0.72	1.00	0.71	0.84
SVM	0.80	0.80	0.81	0.81	0.79	0.79	0.83	0.82	0.90	0.89	69
RF	1.00	0.82	1.00	0.82	1.00	0.82	1.00	0.82	1.00	0.90	10
AdaBoost	0.77	0.77	0.78	0.78	0.76	0.76	0.79	0.79	0.86	0.85	3.5
KNN	0.82	0.72	0.83	0.74	0.79	0.70	0.87	0.78	0.91	0.79	0.063
Stacking	0.81	0.80	0.81	0.80	0.80	0.80	0.81	0.81	0.90	0.89	337

Figure 17 Table of results: training (T) and validation (V) scores

Cross-validation results summarised in Figure 17 show that DT and RF significantly overfitted. They achieved a perfect training score of 1.00 for all metrics indicating they memorised the training set well. However, there were significantly lower validation scores across all metrics. This means they struggled to generalise on unseen data. Furthermore, KNN overfitted but less as the scores are closer together, making it better than DT and RF.

NB cannot identify fake reviews well because it had the lowest overall performance, with 62% accuracy. Nevertheless, it generalised well due to close training and validation scores. It also had the quickest training time of 0.053 seconds.

The best performers were LR, SVM, AdaBoost, and Stacking, as the training and validation scores were very close to each other, meaning they generalised well. Experimenting with different solvers and regularisation strengths for LR produced accuracies that were approximately the same or worse. When done the same for SVM and stacking, it did not improve accuracy, but training time was unrealistically long. The program was stopped manually after two hours highlighting that SVM is computationally expensive and not scalable. Adaboost is chosen for the final model because it provided an optimal compromise between computational time, performance, and generalisation- proven in the next section.

4.2.2 Final Model

4.2.2.1 Hyperparameter Tuning

Despite SVM obtaining the best accuracy, AdaBoost is approximately 20 times faster. It can be shown that AdaBoost can achieve equivalent accuracy with hyperparameter tuning. In Figure 18, Randomised Search was used to identify random combinations of hyperparameters, such as estimator depth, number of trees, and learning rate, for maximising accuracy. Grid Search tests all possible hyperparameter combinations and was not used because it was computationally inefficient.

```
# Defining hyperparameters
parameters = {
    'estimator': [DecisionTreeClassifier(max_depth=i) for i in [1,2,3]], # base learner depth
    'n_estimators': [50, 100, 200, 300], # number of trees
    'learning_rate': [0.01, 0.1, 0.5, 1.0]} # step size

# Initializing AdaBoost
adaboost = AdaBoostClassifier(random_state=42)

# Randomized Searching
random_search = RandomizedSearchCV(
    adaboost, parameters,
    n_iter=10, # try ten different combinations
    cv=3, # 3-fold cross-validation
    scoring='accuracy', # aiming for best accuracy
    n_jobs=-1, # parallel processing to speed up
    random_state=42)
random_search.fit(X_train, y_train)
print("Best AdaBoost Parameters:", random_search.best_params_)
```

```
Best AdaBoost Parameters: {'n_estimators': 300, 'learning_rate': 0.5, 'estimator': DecisionTreeClassifier(max_depth=3
```

Figure 18 Hyperparameter Tuning using Random Search

The best parameters were 'n_estimators': 300, 'learning_rate': 0.5, 'estimator': DecisionTreeClassifier(max_depth=3). In Figure 19, accuracy of 0.81 was achieved which is approximately equivalent to SVM accuracy but significantly faster.

```python
# Pipeline for scaling and Tuned AdaBoost classifier
pipeline_AdaBoost_tuned = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('AdaBoost', AdaBoostClassifier(
        estimator=DecisionTreeClassifier(max_depth=3),
        n_estimators=300,
        learning_rate=0.5,
        algorithm='SAMME',
        random_state=42))])

cross_validation(pipeline_AdaBoost_tuned, X_train, y_train, cv=5, model_name="AdaBoost (Tuned)")
```

```
AdaBoost (Tuned)  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.824146          0.814562
       f1     0.824931          0.815297
precision     0.822673          0.813251
   recall     0.827222          0.817469
  roc_auc     0.910698          0.902340
Mean Fit Time: 50.38752765655518
```

Figure 19 Cross-validation of tuned AdaBoost

4.2.2.2 Test set evaluation

Trained optimised AdaBoost was tested using the unseen test set to gauge its final performance, the results are presented in Figure 20.

Model	Accuracy	F1	Precision	Recall	ROC_AUC	Time (s)
AdaBoost	0.82	0.82	0.81	0.83	0.91	43.5

Figure 20 Final model performance

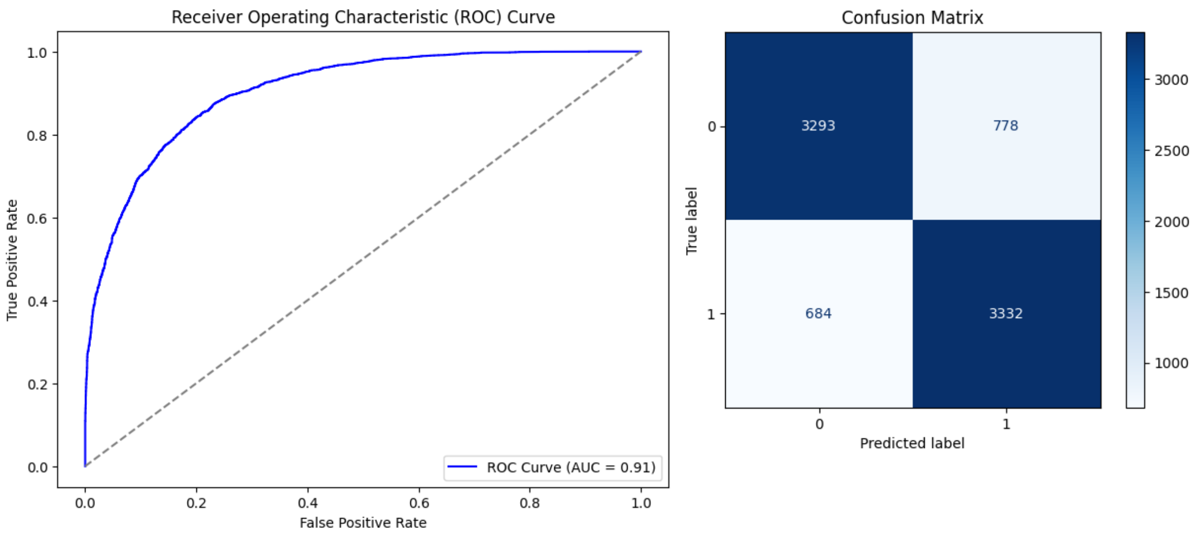

Figure 21 ROC Curve (left) and Confusion Matrix (right)

From the confusion matrix, performance using the test set shows that the model can strongly and correctly classify both real and fake reviews. The model accurately identified 3332 fake reviews (Class 1) and 3293 real reviews (Class 0). However, there were a significant number of misclassifications: 684 real reviews were incorrectly labelled as fake, and 778 fake reviews were misclassified as real.

The ROC curve is the True Positive Rate against the False Positive Rate; this demonstrates how well the model separates real and fake reviews. The model achieved an impressive AUC of 0.91, effectively distinguishing between real and fake reviews. In Figure 21, the curve is positioned far from the diagonal line (AUC = 0.5), confirming strong classification.

4.3 Conclusion and Future Work

In conclusion, all five project objectives have been achieved successfully. Cross-validation ensured informed selection of the best model based on performance metrics, while test data evaluation rigorously tested its reliability for fake review identification. However, the most challenging part of this project was feature extraction. Text data required complex NLP techniques to create useful features, which was time-consuming and computationally demanding. This challenge was amplified by the limitation of using the free version of Google Collab with restricted computational resources.

To remedy this challenge, one future research direction could be using more RAM and more powerful CPU and GPU. This will allow more complex features to be processed, like using TF-IDF to consider n-grams which assess importance of words or phrases in a review. This will enable models to detect more subtle patterns in fake reviews and improve identification accuracy.

Other future research directions include using transformer models to compare performance with my results, like BERT or RoBERTa which are neural network models that understands text context better. Also, more features like user metadata (like review history and patterns) and more training dataset (more varied comment styles) would improve the model's identification.

References

Abdulqader, M., Namoun, A. and Alsaawy, Y., 2022. Fake online reviews: a unified detection model using deception theories. *IEEE Access*, 10, 128622–128655.

Alsubari, S., Shelke, M. and Deshmukh, S., 2020. Fake reviews identification based on deep computational linguistic features. *International Journal of Advanced Science and Technology*, 29(7), 3846–3856.

Alsubari, S., Deshmukh, S., Alqarni, A., Alsharif, N., H, T., Alsaade, F. and Khalaf, O., 2021. Data analytics for the identification of fake reviews using supervised learning. *Computers, Materials & Continua*, 70(2), 3189–3204.

Competition and Markets Authority, 2020. *Online reviews: full publication update history*. United Kingdom: UK.GOV. Available from: https://www.gov.uk/cma-cases/online-reviews#full-publication-update-history [Accessed 21 February 2025].

Elmogy, A.M., Tariq, U., Mohammed, A. and Ibrahim, A., 2021. Fake reviews detection using supervised machine learning. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 12(1). Available from: https://thesai.org/Publications/ViewPaper?Volume=12&Issue=1&Code=IJACSA&SerialNo=69 [Accessed 18 March 2025].

Jain, P.K., Pamula, R. and Ansari, S., 2021. A supervised machine learning approach for the credibility assessment of user-generated content. *Wireless Personal Communications*, 118(4), 2469–2485.

K P, R., Prabu Nallasivam, M., Rajan C, S.G., Paul P S, S., S, H.K. and V S, D., 2024. Detection of fake hotel reviews using ANFIS and natural language processing techniques. *In:* 2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT). *7th International Conference on Circuit Power and Computing Technologies (ICCPCT)*, 265–269. Available from: https://ieeexplore.ieee.org/abstract/document/10672838 [Accessed 17 February 2025].

Salminen, J., 2025. Fake Reviews Dataset. Available from: https://osf.io/tyue9

Shan, G., Zhou, L. and Zhang, D., 2021. From conflicts and confusion to doubts: examining review inconsistency for fake review detection. *Decision Support Systems*, 144, 113513.

Appendix A Detailed Dataset Overview

	category	rating	label	text_
40412	Clothing_Shoes_and_Jewelry_5	5.0	CG	OMG, First reason why I chose this style is be...
40413	Clothing_Shoes_and_Jewelry_5	5.0	OR	The cold damp fog tries to grab at my ankles a...
40414	Clothing_Shoes_and_Jewelry_5	4.0	CG	First off I returned this pair. The quality is...
40415	Clothing_Shoes_and_Jewelry_5	5.0	OR	I'd consider myself an advanced non-profession...
40416	Clothing_Shoes_and_Jewelry_5	3.0	CG	When I saw this ring, I thought it was very be...
40417	Clothing_Shoes_and_Jewelry_5	5.0	OR	I cannot thank my boss enough for recommending...
40418	Clothing_Shoes_and_Jewelry_5	2.0	CG	This bag weighs exactly 10pounds (I wear an XL...
40419	Clothing_Shoes_and_Jewelry_5	5.0	OR	Sometimes it is so hard to find a loose comfor...
40420	Clothing_Shoes_and_Jewelry_5	5.0	CG	I just bought these locally and they are the b...
40421	Clothing_Shoes_and_Jewelry_5	2.0	OR	I'm a 36B, which means that I don't need any h...
40422	Clothing_Shoes_and_Jewelry_5	4.0	CG	I wore this from 4pm to 9pm and it was perfect...
40423	Clothing_Shoes_and_Jewelry_5	4.0	OR	This is a classy looking watch. I don't get m...
40424	Clothing_Shoes_and_Jewelry_5	3.0	CG	I kind of feel giving it a 3 star because it's...
40425	Clothing_Shoes_and_Jewelry_5	5.0	OR	The stated dimensions on the description are o...
40426	Clothing_Shoes_and_Jewelry_5	5.0	CG	Overall, I love this hat!\n\nSize/Color: 9.5/...
40427	Clothing_Shoes_and_Jewelry_5	4.0	OR	I had read some reviews saying that this bra r...
40428	Clothing_Shoes_and_Jewelry_5	5.0	CG	I wasn't sure exactly what it would be. It is ...
40429	Clothing_Shoes_and_Jewelry_5	2.0	OR	You can wear the hood by itself, wear it with ...
40430	Clothing_Shoes_and_Jewelry_5	1.0	CG	I liked nothing about this dress. The only rea...
40431	Clothing_Shoes_and_Jewelry_5	5.0	OR	I work in the wedding industry and have to wor...

Figure A.1 Last 20 instances of the dataset

Appendix B Features Extracted using NLP

VERB	Verb
DET	Determiner
NOUN	Noun
CONJ	Conjunction
ADV	Adverb
ADJ	Adjective
PRON	Pronoun
ADP	Adposition
PRT	Particle
NUM	Numeral
X	Other

Figure B.1 Linguistic Feature: POS Tagging

advcl	Adverbial clause modifier	xcomp	Open clausal complement
nsubj	Nominal subject	auxpass	Auxiliary passive
advmod	Adverbial modifier	mark	Marker
relcl	Relative clause modifier	nsubjpass	Nominal subject (passive)
cop	Copula	csubj	Clausal subject
prd	Predicate	npadvmod	Nominal adverbial modifier
cc	Coordinating conjunction	intj	Interjection
conj	Conjunct	dative	Dative
ROOT	Root	compound	Compound modifier
dobj	Direct object	dep	Unspecified dependency
det	Determiner	acln	Adjectival clausal modifier
amod	Adjectival modifier	nmod	Nominal modifier
prep	Prepositional modifier	predet	Predeterminer
pobj	Object of preposition	appos	Appositional modifier
aux	Auxiliary	prt	Particle
poss	Possessive modifier	expl	Expletive
parataxis	Parataxis	preconj	Preconjunction
pcomp	Prepositional complement	quantmod	Quantifier modifier
attr	Attribute	csubjpass	Clausal subject (passive)
nummod	Numerical modifier	agent	Agent
neg	Negation modifier	case	Case
ccomp	Clausal complement	meta	Meta
acomp	Adjectival complement		

Figure B.2 Linguistic Feature: Dependency Parsing

neg	negative sentiment
neu	neutral sentiment
pos	positive sentiment

Figure B.3 Sentiment Feature: Sentiment Analysis

Appendix C Detailed Model Performance

```
# Pipeline for scaling and LogisticRegression classifier
pipeline_LR = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('LR', LogisticRegression(random_state=42, solver='saga'))]) # 'saga', good for large datasets

cross_validation(pipeline_LR, X_train, y_train, cv=5, model_name="Logistic Regression")
```

```
Logistic Regression  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.778868          0.777091
       f1     0.782788          0.781203
precision     0.770424          0.768434
   recall     0.795556          0.794444
  roc_auc     0.870728          0.869786
Mean Fit Time: 2.5173527240753173
```

Figure C.1 Logistic Regression

```
# Pipeline for scaling and multinomial Naive Bayes classifier
pipeline_NB = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))), # needs non-negative values so MinMaxScaler
    ('NB', MultinomialNB())])

cross_validation(pipeline_NB, X_train, y_train, cv=5, model_name="Naive Bayes")
```

```
Naive Bayes  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.622886          0.621178
       f1     0.676382          0.675031
precision     0.593112          0.591828
   recall     0.786852          0.785494
  roc_auc     0.675502          0.674826
Mean Fit Time: 0.05321755409240723
```

Figure C.2 Naïve Bayes

```
# Pipeline for scaling and decision tree classifier
pipeline_DT = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('DT', DecisionTreeClassifier(random_state=42))])

cross_validation(pipeline_DT, X_train, y_train, cv=5, model_name="Decision Tree")
```

```
Decision Tree  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy          1.0          0.712660
       f1          1.0          0.713656
precision          1.0          0.712386
   recall          1.0          0.715000
  roc_auc          1.0          0.712656
Mean Fit Time: 0.8366841793060302
```

Figure C.3 Decision Tree

```
# Pipeline for scaling and C-Support Vector classifier
pipeline_SVM = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('SVM', SVC(random_state=42))]) # kernel='rbf', Radial Basis Function, by default

cross_validation(pipeline_SVM, X_train, y_train, cv=5, model_name="SVM")
```

```
SVM  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.804808          0.800897
       f1     0.809261          0.805437
precision     0.792541          0.788770
   recall     0.826713          0.822840
  roc_auc     0.895593          0.892570
Mean Fit Time: 68.79906368255615
```

Figure C.4 Support Vector Machine

```
# Pipeline for scaling and random forest classifier
pipeline_RF = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('RF', RandomForestClassifier(random_state=42))])

cross_validation(pipeline_RF, X_train, y_train, cv=5, model_name="Random Forest")
```

```
Random Forest  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.999992          0.820498
       f1     0.999992          0.820904
precision     0.999985          0.820473
   recall     1.000000          0.821358
  roc_auc     1.000000          0.902450
Mean Fit Time: 10.43918571472168
```

Figure C.5 Random Forest

```
# Pipeline for scaling and AdaBoost classifier
pipeline_AdaBoost = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('AdaBoost', AdaBoostClassifier(random_state=42))])

cross_validation(pipeline_AdaBoost, X_train, y_train, cv=5, model_name="AdaBoost")
```

```
AdaBoost  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.771595          0.770134
       f1     0.776848          0.775863
precision     0.760883          0.758580
   recall     0.793951          0.794321
  roc_auc     0.855928          0.853588
Mean Fit Time: 3.5297318935394286
```

Figure C.6 Adaptive Boosting

```
# Pipeline for scaling and k-nearest neighbours classifier
pipeline_KNN = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('KNN', KNeighborsClassifier())]) # n_neighbors=5 by default

cross_validation(pipeline_KNN, X_train, y_train, cv=5, model_name="k-nearest neighbours")
```

```
k-nearest neighbours  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.821348          0.720390
       f1     0.829562          0.737144
precision     0.794326          0.696605
   recall     0.868071          0.782716
  roc_auc     0.906164          0.789472
Mean Fit Time: 0.06344928741455078
```

Figure C.7 K-nearest Neighbours

```
base_models = [
    ('LR', LogisticRegression(random_state=42, solver='saga')),
    ('SVM', SVC(random_state=42))]

# Defining Stacking Classifier
stack = StackingClassifier(estimators=base_models, final_estimator=LogisticRegression())

pipeline_stack = Pipeline([
    ('scaler', MinMaxScaler(feature_range=(0, 1))),
    ('stack', stack)])

cross_validation(pipeline_stack, X_train, y_train, cv=5, model_name="Stacking Classifier")
```

```
Stacking Classifier  Cross-Validation Results:
   Metric  Train Score  Validation Score
 accuracy     0.805681          0.801360
       f1     0.807591          0.803247
precision     0.801078          0.797039
   recall     0.814213          0.809568
  roc_auc     0.895500          0.892534
Mean Fit Time: 336.71405377388
```

Figure C.8 Stacking

Appendix D Dependencies

```
scikit-learn: 1.6.1
NLTK: 3.9.1
SpaCy: 3.8.5
Pandas: 2.2.2
NumPy: 2.0.2
Python: 3.11.12 (main, Apr  9 2025, 08:55:54) [GCC 11.4.0]
```

Figure D.1 Versions of libraries or tools used for this project