🕵️ Fake Job Posting Detection

A project to detect fraudulent job postings using Natural Language Processing (NLP) and Machine Learning techniques.

🎯 Objective

To build a classifier that identifies fake job postings based on the text content in the job advertisement.

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

# Load the dataset
job = pd.read_csv(r"F:\CSV Files\fake_job_postings.csv")
job.head()
```

Out[1]:

	job_id	title	location	department	salary_range	company_profile	descripti
0	1	Marketing Intern	US, NY, New York	Marketing	NaN	We're Food52, and we've created a groundbreaki...	Food52, a fa growing, James Bea Award-winr
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	NaN	90 Seconds, the worlds Cloud Video Production ...	Organised - Focuse Vibrant - Awesome!I you
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever	NaN	NaN	Valor Services provides Workforce Solutions th...	Our client, located Houston, is activ se
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	NaN	Our passion for improving quality of life thro...	THE COMPANY: ES – Environmen Systems Rese
4	5	Bill Review Manager	US, FL, Fort Worth	NaN	NaN	SpotSource Solutions LLC is a Global Human Cap...	JOB TITL Itemization Revi ManagerLOCATION

🔧 Data Preprocessing

In [2]:

```
# Convert to string to avoid type issues
text_columns = ["company_profile", "description", "requirements", "benefits","rec
for col in text_columns:
    job[col] = job[col].astype(str)

# Combine text into one column
job["X"] = job["company_profile"] + " " + job["description"] + " " + job["require
job.rename(columns={'fraudulent':'Y'}, inplace=True)
job = job.loc[:, ['X', 'Y']]
job['X'] = job['X'].str.lower()
job.head()
```

Out[2]:

	X	Y
0	we're food52, and we've created a groundbreaki...	0
1	90 seconds, the worlds cloud video production ...	0
2	valor services provides workforce solutions th...	0
3	our passion for improving quality of life thro...	0
4	spotsource solutions llc is a global human cap...	0

🧹 Text Cleaning & Tokenization

In [3]:
```
import nltk
from nltk.corpus import stopwords
import string

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def text_process(text):
    no_punc = ''.join([char for char in text if char not in string.punctuation])
    return [word for word in no_punc.split() if word not in stop_words]

job['X'] = job['X'].astype(str)
job['X_clean'] = job['X'].apply(text_process)
job['X_clean'].head()
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[3]:
```
0    [food52, weve, created, groundbreaking, awardw...
1    [90, seconds, worlds, cloud, video, production...
2    [valor, services, provides, workforce, solutio...
3    [passion, improving, quality, life, geography,...
4    [spotsource, solutions, llc, global, human, ca...
Name: X_clean, dtype: object
```

🧠 Feature Extraction using Bag of Words

In [4]:
```
from sklearn.feature_extraction.text import CountVectorizer

bow_transformer = CountVectorizer(analyzer=text_process)
X = bow_transformer.fit_transform(job['X'])

X.shape()
```

Out[4]:
```
(17880, 171575)
```

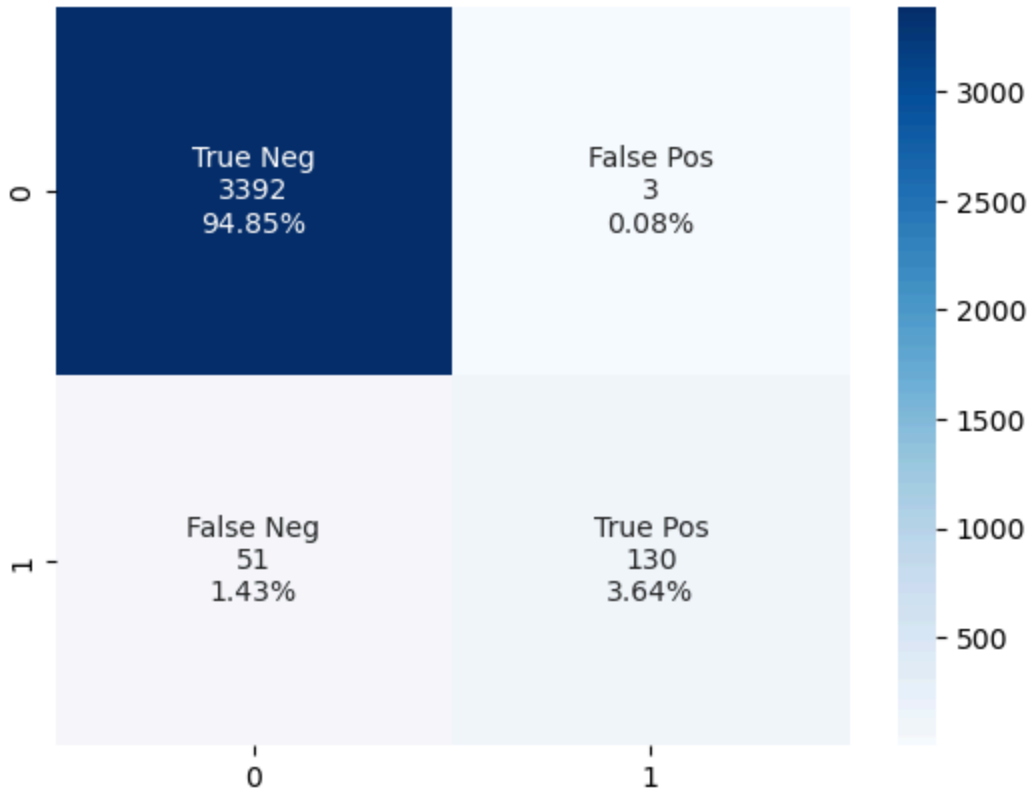
🤖 Model Building & Evaluation

In [5]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

y = job['Y']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s

# Logistic Regression
log_model = LogisticRegression()
log_model.fit(X_train, y_train)
pred_log = log_model.predict(X_test)

print("🔍 Logistic Regression Performance:")
print(confusion_matrix(y_test, pred_log))
print(classification_report(y_test, pred_log))


# Decision Tree Classifier
dt_model = DecisionTreeClassifier(criterion='entropy', max_depth=10, min_samples_
dt_model.fit(X_train, y_train)
pred_dt = dt_model.predict(X_test)

print("🌲 Decision Tree Performance:")
print(confusion_matrix(y_test, pred_dt))
print(classification_report(y_test, pred_dt))
```

```
🔍 Logistic Regression Performance:
[[3392    3]
 [  51  130]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      3395
           1       0.98      0.72      0.83       181

    accuracy                           0.98      3576
   macro avg       0.98      0.86      0.91      3576
weighted avg       0.98      0.98      0.98      3576

🎄 Decision Tree Performance:
[[3378   17]
 [  99   82]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      3395
           1       0.83      0.45      0.59       181

    accuracy                           0.97      3576
   macro avg       0.90      0.72      0.78      3576
weighted avg       0.96      0.97      0.96      3576
```

In [9]:
```
cf_matrix = confusion_matrix(y_test, pred_log)
import numpy as np;
import seaborn as sns
group_names = ["True Neg","False Pos", "False Neg", "True Pos"]
group_counts = ["{0:0.0f}".format(value) for value in
                cf_matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]
labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues');
```



✅ Conclusion

- Logistic Regression performed better in detecting fake jobs with high accuracy.
- Text preprocessing and Bag of Words significantly helped in feature extraction.

In [ ]:
```
```