"Fake Job Postings Detection Using Machine Learning”

CONCEPT OF THIS PROJECT

The concept of this project is to detect fraudulent job postings using machine learning and natural language processing techniques. It analyzes the text of job titles and descriptions, converts them into numerical features using TF-IDF, and trains a Logistic Regression model to classify jobs as real or fake. By learning patterns from labeled data, the model can predict whether a new job posting is likely to be a scam, helping users and job platforms filter out fake listings and improve trust and safety.

COMPONENTS USED AND CODE

This project uses Python programming along with libraries like Pandas and NumPy for data handling, Matplotlib and Seaborn for visualization, and scikit-learn for machine learning. It uses TF-IDF Vectorizer for converting job text data into numerical format, Logistic Regression for classification, and evaluation metrics like accuracy, precision, recall, and F1-score to measure model performance. The dataset fake_job_postings.csv is used as input, containing job titles, descriptions, and labels indicating whether the job is real or fake.

```
mport pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
df = pd.read_csv("fake_job_postings.csv")
df = df.dropna(subset=['description', 'title'])
df = df[['title', 'description', 'fraudulent']]
df['text'] = df['title'] + " " + df['description']
df['text'] = df['text'].str.lower().str.replace(r'[^a-zA-Z0-9 ]', '', regex=True)
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X = tfidf.fit_transform(df['text']).toarray()
y = df['fraudulent']
X_train, X_test, y_train, y_test = train_test_split(X, y.       test_size =0.2,rando
m_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
def predict_job_fraud(title, description):
text = title + " " + description
text = text.lower().replace(r'[^a-zA-Z0-9 ]', '')
text_vectorized = tfidf.transform([text]).toarray()
prediction = model.predict(text_vectorized)
return "Fake" if prediction[0] == 1 else "Real”
```

STEP 1: IMPORTING REQUIRED LIBRARY

```
import pandas as pd, numpy as np

import matplotlib.pyplot as plt, seaborn as sns

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, accuracy_score
```

Imports tools for data handling (pandas, numpy), visualization (matplotlib, seaborn), and machine learning (scikit-learn).

Step 2: Load the Dataset

```
df = pd.read_csv("fake_job_postings.csv")
```

Loads the dataset (CSV file) containing job posting data.

Step 3: Clean Missing Values

```
df = df.dropna(subset=['description', 'title'])
```

Drops rows where the title or description is missing (important for prediction).

Step 4: Select Important Columns

```
df = df[['title', 'description', 'fraudulent']]
```

Keeps only the necessary columns for analysis:

title, description: job details

fraudulent: target label (0 = Real, 1 = Fake)

Step 5: Combine Title & Description

```
df['text'] = df['title'] + " " + df['description']
```

Combines job title and description into one text column for better feature extraction.

Step 6: Text Preprocessing

```
df['text'] = df['text'].str.lower().str.replace(r'[^a-zA-Z0-9 ]', '', regex=True)
```

Converts text to lowercase and removes special characters to clean the data.

Step 7: Vectorization using TF-IDF

```
tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
X = tfidf.fit_transform(df['text']).toarray()
y = df['fraudulent']
```

Converts text into numeric format using TF-IDF (Term Frequency - Inverse Document Frequency).

Limits to 5000 most important words.

X is the feature matrix; y is the target label.

Step 8: Split Dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Splits data into:

80% for training

20% for testing

random_state=42 ensures results are repeatable.

Step 9: Model Training

```
model = LogisticRegression()

model.fit(X_train, y_train)
```

Trains a Logistic Regression model using the training data.

Step 10: Predictions and Evaluation

```
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

Makes predictions on the test data.

Prints:

Accuracy: Overall correctness of model.

Classification report: Precision, recall, F1-score for real and fake jobs.

Prediction Function

```
def predict_job_fraud(title, description):
    text = title + " " + description
    text = text.lower().replace(r'[^a-zA-Z0-9 ]', '')
    text_vectorized = tfidf.transform([text]).toarray()
    prediction = model.predict(text_vectorized)
    return "Fake" if prediction[0] == 1 else "Real"
```

This function takes a new job title and description and:

Preprocesses the text

Vectorizes it using TF-IDF

Predicts if it's "Fake" or "Real" using the trained model.

SAMPLE OUTPUT

Accuracy: 0.96

	precision	recall	f1-score	support
0	0.97	0.99	0.98	3191
1	0.85	0.64	0.73	207
accuracy			0.96	3398
macro avg	0.91	0.82	0.85	3398
weighted avg	0.96	0.96	0.96	3398