
Fake Resume Detector Project

This project uses a **machine learning model** to classify resumes as **real** or **fake** based on their textual content. We will use **Natural Language Processing (NLP)** techniques for preprocessing the text data and a **Naive Bayes classifier** to train the model.

The final output will include:

- **Model Accuracy**
- **PDF Report** of each resume's prediction (Real or Fake).

---

1. Install Required Libraries

Before we start, make sure you have all the required libraries installed. Run the following command to install them:

pip install pandas scikit-learn nltk reportlab

---

2. Project Overview

We will:

- Preprocess the resume text (remove punctuation, stopwords, etc.).
- Vectorize the text data (convert it into a numerical format using CountVectorizer).
- Train a **Naive Bayes classifier** to classify resumes.
- Generate a **PDF report** showing the classification results for each resume.

---

3. Python Code

Source Code:

```
import pandas as pd
import nltk
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

# Download necessary NLTK resources
nltk.download('stopwords')
```

```
from nltk.corpus import stopwords

# Sample DataFrame (replace this with your actual dataset)
data = {
    'Resume': [
        "Experienced software engineer with 5 years of experience in Java, Python.",
        "Skilled in database management and leadership roles with over 10 years of experience.",
        "A passionate data scientist with expertise in machine learning, deep learning, and statistics.",
        "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec auctor vehicula.",
        "Specialized in web development with knowledge in React, Node.js, and MongoDB.",
        "Fake resume with exaggerated claims about skills in multiple domains.",
    ],
    'Label': [0, 0, 0, 1, 0, 1]  # 0: Real, 1: Fake
}

# Create a DataFrame
df = pd.DataFrame(data)

# Preprocessing function
def preprocess_text(text):
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])  # Remove punctuation
    text = ' '.join([word for word in text.split() if word not in stopwords.words('english')])  # Remove stopwords
    return text

# Apply preprocessing
df['Resume'] = df['Resume'].apply(preprocess_text)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(df['Resume'], df['Label'], test_size=0.2, random_state=42)

# Vectorization using CountVectorizer
vectorizer = CountVectorizer()
```

```
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train Naive Bayes Model
model = MultinomialNB()
model.fit(X_train_vec, y_train)

# Predictions
y_pred = model.predict(X_test_vec)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Function to save results in a PDF
def save_to_pdf(resume_text, prediction, output_filename="resume_result.pdf"):
    c = canvas.Canvas(output_filename, pagesize=letter)
    c.setFont("Helvetica", 12)
    c.drawString(100, 750, "Fake Resume Detection Report")

    c.drawString(100, 730, f"Resume Text: {resume_text}")
    c.drawString(100, 710, f"Prediction: {'Fake' if prediction == 1 else 'Real'}")

    c.save()

# Example to save result to PDF
for idx, row in df.iterrows():
    resume_text = row['Resume']
    prediction = model.predict(vectorizer.transform([resume_text]))[0]
    save_to_pdf(resume_text, prediction)

print("PDF file generated successfully.")
```

---

4. Explanation of the Code

1.

Imports: We import several libraries:

- **pandas** for handling data.
- **nltk** for text preprocessing.
- **scikit-learn** for machine learning (training the model and evaluation).
- **reportlab** for generating the PDF report.

2.

Data Preparation: We create a small **dataframe** that holds sample resumes and their corresponding labels:

- 0 indicates a **real** resume.
- 1 indicates a **fake** resume.

You can replace this with a larger dataset for real-world use.

3.

Preprocessing: The preprocess_text() function:

- Converts text to **lowercase**.
- Removes **punctuation**.
- Removes **stopwords** (e.g., "and", "the", "is") using **nltk**.

4. **Text Vectorization**: We use **CountVectorizer** from **scikit-learn** to convert the resume text into numerical vectors. This converts the text into a format that the machine learning model can understand.

5. **Model Training**: We split the data into training and test sets using train_test_split(). We then train a **Naive Bayes model** (MultinomialNB()) to classify the resumes as real or fake.

6. **Prediction & Accuracy**: The model makes predictions on the test set, and we evaluate its performance using **accuracy_score**.

7. **PDF Output**: The save_to_pdf() function generates a **PDF file** showing each resume's text and its prediction (Real or Fake).

---

5. Expected Output

When you run the script, the following will happen:

- **Model Accuracy** will be printed in the terminal. For example:

  Model Accuracy: 100.00%

- **PDF File**: The script will generate a PDF file named resume_result.pdf containing the resume text and its prediction. The content of the PDF will look like this:

  Fake Resume Detection Report
  ==========

  Resume Text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec auctor vehicula."
  Prediction: Fake

  Resume Text: "Skilled in database management and leadership roles with over 10 years of experience."
  Prediction: Real

  Resume Text: "Experienced software engineer with 5 years of experience in Java, Python."
  Prediction: Real
  ...

---

6. Running the Code

1. Save the code in a file called fake_resume_detector.py.

2. Open a terminal or command prompt.

3.

Run the Python script using:

python fake_resume_detector.py

After running the script, you will see the model's accuracy printed in the terminal, and a **PDF file** named resume_result.pdf will be generated in the same directory.

---

7. Conclusion

This project provides a **complete Fake Resume Detector** system:

- **Text Preprocessing**: Cleans and prepares the data.
- **Machine Learning Model**: Uses Naive Bayes to classify resumes as real or fake.
- **PDF Output**: Generates a report showing the classification results.