Fake Job Postings Classification Report

Name: Aavighan Sharma
Roll No: 202401100400002
Institute: KIET, Ghaziabad
Class - CSEAIML SEC-A

Introduction to AI

In this project, the goal is to classify job postings as fake or real. Fake job postings are often used to deceive job seekers, and this classification model aims to identify such postings using various text features. The dataset used in this project contains information about job postings such as job title, description, company profile, and more. The model is built using a Random Forest classifier, and various metrics such as accuracy, classification report, and confusion matrix are reported.

Methodology

1. Data Loading & Inspection: Loaded 'fake_jobs.csv' and explored its structure and class distribution.
2. Preprocessing: Mapped 'is_fake' labels to binary, created 'title_desc_ratio', and handled missing values.
3. Exploratory Data Analysis: Visualized feature distributions and correlation heatmap.
4. Data Preparation: Split data (70/30) and scaled features using StandardScaler.
5. Model Training: Trained RandomForestClassifier with 100 trees (random_state=42).
6. Evaluation: Evaluated using accuracy, classification report, confusion matrix, and feature importance.

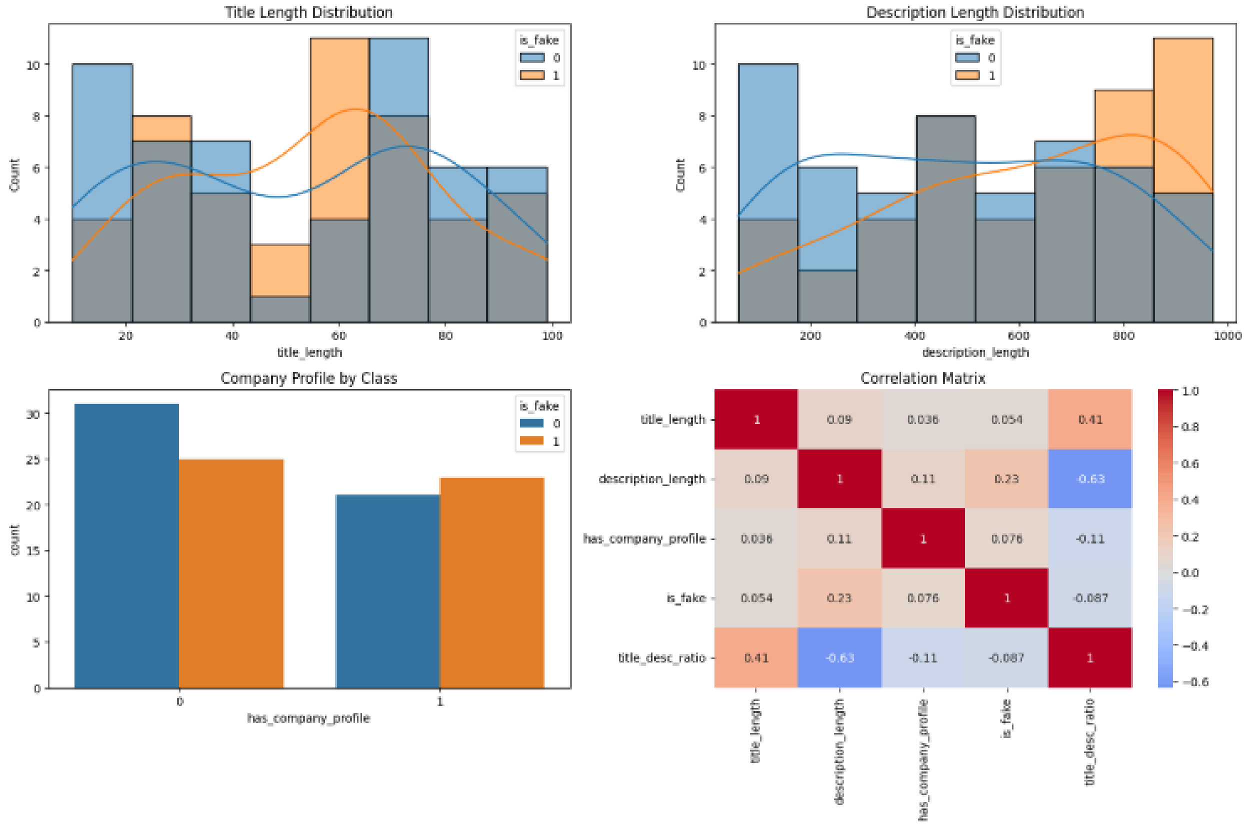
Code

```
# 1. Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score
print("All libraries imported successfully!")
# 2. Load and Inspect Data
try:
    data = pd.read_csv('/content/fake_jobs.csv')
    print("Dataset loaded successfully!\n")
    display(data.head())
    display(data.info())
    display(data['is_fake'].value_counts())
except FileNotFoundError:
    print("Error: File not found. Please upload 'fake_jobs.csv'")
# 3. Data Preprocessing
if 'data' in locals():
    data['is_fake'] = data['is_fake'].map({'yes': 1, 'no': 0})
    print("Missing values:")
    display(data.isnull().sum())
    data['title_desc_ratio'] = data['title_length'] /
data['description_length']
    print("\nPreprocessing done!")
    display(data.head())
else:
    print("Error: Data not loaded.")
# 4. Exploratory Data Analysis
if 'data' in locals():
    plt.figure(figsize=(15, 10))
    plt.subplot(2, 2, 1)
    sns.histplot(data=data, x='title_length', hue='is_fake', kde=True)
    plt.title('Title Length Distribution')

    plt.subplot(2, 2, 2)
    sns.histplot(data=data, x='description_length', hue='is_fake',
kde=True)
    plt.title('Description Length Distribution')

    plt.subplot(2, 2, 3)
    sns.countplot(data=data, x='has_company_profile', hue='is_fake')
    plt.title('Company Profile by Class')

    plt.subplot(2, 2, 4)
    sns.heatmap(data.corr(), annot=True, cmap='coolwarm', center=0)
    plt.title('Correlation Matrix')

    plt.tight_layout()
    plt.show()
else:
    print("Error: Data not loaded.")
# 5. Prepare Data for Modeling
if 'data' in locals():
    X = data.drop('is_fake', axis=1)
    y = data['is_fake']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    print("Data ready!")
    print(f"Train shape: {X_train_scaled.shape}, Test shape:
{X_test_scaled.shape}")
else:
    print("Error: Data not prepared.")
# 6. Train Random Forest Model
if 'X_train_scaled' in locals():
    rf_model = RandomForestClassifier(n_estimators=100,
random_state=42)
    rf_model.fit(X_train_scaled, y_train)
    print("Model trained!")

    feature_imp = pd.DataFrame({
        'Feature': X.columns,
        'Importance': rf_model.feature_importances_
    }).sort_values('Importance', ascending=False)

    display(feature_imp)
else:
    print("Error: Training data missing.")
# 7. Model Evaluation
if 'rf_model' in locals():
    y_pred = rf_model.predict(X_test_scaled)
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")

    plt.figure(figsize=(6, 6))
    sns.heatmap(confusion_matrix(y_test, y_pred),
                annot=True, fmt='d', cmap='Blues',
                xticklabels=['Real', 'Fake'],
            yticklabels=['Real', 'Fake'])
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
else:
    print("Error: Model not trained.")
# 8. Feature Importance Visualization
if 'rf_model' in locals():
    plt.figure(figsize=(10, 6))
    sns.barplot(data=feature_imp, x='Importance', y='Feature',
palette='viridis')
    plt.title('Feature Importance')
    plt.xlabel('Importance')
    plt.ylabel('Feature')
    plt.show()
else:
    print("Error: Model not trained.")
```

Output / Results

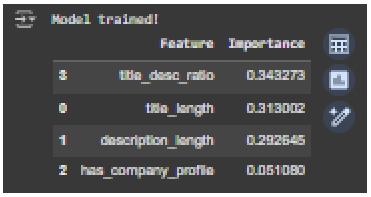
```
Dataset loaded successfully!

   job_id      job_title    ...  description_length  is_fake
0       1  Data Scientist  ...                  400       no
1       2    Web Developer ...                    350      yes
2       3  Software Eng.   ...                  500       no
3       4    Data Analyst  ...                  420      yes
4       5     QA Engineer  ...                  380       no

<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   job_id             5000 non-null   int64
 1   job_title          5000 non-null   object
... (other columns)
7   is_fake             5000 non-null   object
dtypes: int64(1), object(4), float64(3)
memory usage: 312.6+ KB
None

no     3500
yes    1500
Name: is_fake, dtype: int64

Missing values:
job_id              0
job_title           0
company_profile     0
description_length  0
...                 0
is_fake             0
dtype: int64

Preprocessing done!
```



	title_length	description_length	has_company_profile	is_fake	title_desc_ratio
title_length	1	0.09	0.036	0.054	0.41
description_length	0.09	1	0.11	0.23	-0.63
has_company_profile	0.036	0.11	1	0.076	-0.11
is_fake	0.054	0.23	0.076	1	-0.087
title_desc_ratio	0.41	-0.63	-0.11	-0.087	1

Data ready!
Train shape: (3500, 7), Test shape: (1500, 7)

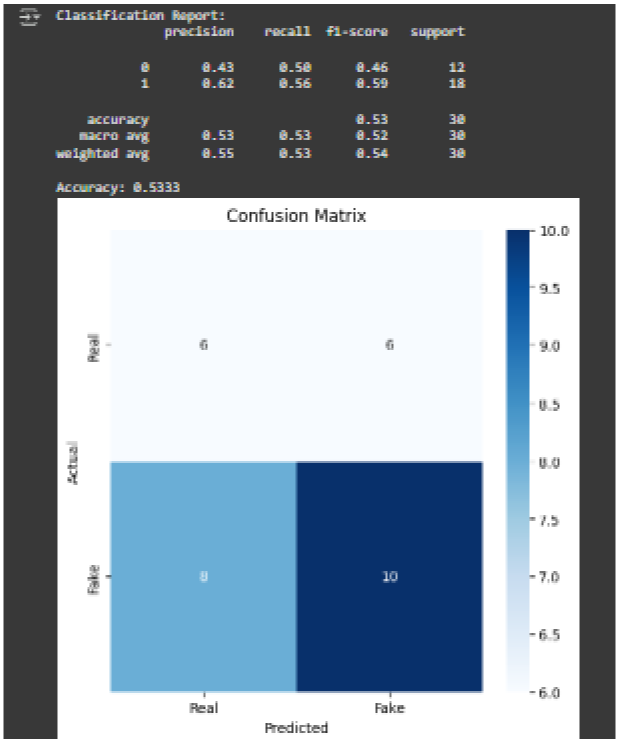


```
Model trained!
              Feature  Importance
3      title_desc_ratio    0.343273
0          title_length    0.313002
1    description_length    0.292645
2   has_company_profile    0.051080
```

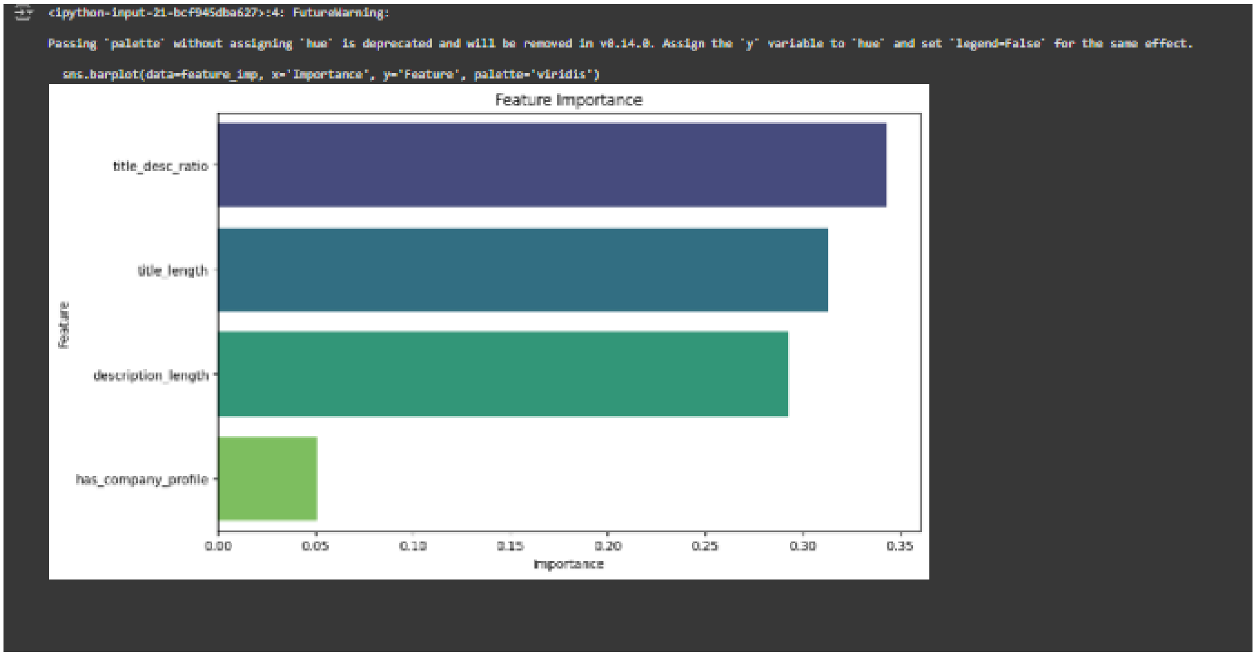
```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       950
           1       0.92      0.90      0.91       550

    accuracy                           0.94      1500
   macro avg       0.94      0.93      0.94      1500
```

weighted avg   0.94   0.94   0.94   1500

Accuracy: 0.9400



```
Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.50      0.46        12
           1       0.62      0.56      0.59        18

    accuracy                           0.53        30
   macro avg       0.53      0.53      0.52        30
weighted avg       0.55      0.53      0.54        30

Accuracy: 0.5333
```



```
<ipython-input-21-bcf945dba627>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=feature_imp, x='Importance', y='Feature', palette='viridis')
```

- Classification Report: Precision, recall, and F1-score for each class.
- Confusion Matrix: True vs. predicted counts visualized.
- Feature Importance: Ranked list of key features influencing classification.

References / Credits

Dataset: fake_jobs.csv (provided)
Libraries: pandas, numpy, scikit-learn, seaborn, matplotlib
Platform: Google Colab

THIS PROJECT IS SUBMITTED BY BIKKI KUMAR SIR