Project:-Real/Fake Job Posting Prediction Using Random Forest, Naive Bayes, and Decision Tree Methods

Introduction/Motivation/Problem Definition

The increasing prevalence of online job platforms has led to a surge in job postings. However, amidst genuine opportunities, there is a growing concern about the rise of fake job postings. These deceptive listings not only waste job seekers' time but can also lead to financial scams and identity theft. The motivation behind this project is to develop a robust predictive model that can differentiate between real and fake job postings, thereby enhancing the security and trustworthiness of online job platforms.

Prior Work

Several studies have addressed the challenge of detecting fake job postings, with notable contributions in the field. Vidros et al. [1] made major contributions to properly identifying frauds in the online procedure.The author discusses the severity and distinct characteristics of employment scam, which is a primarily text-based problem that presents several peculiarities.Authors conclude that a fully-fledged solution to online recruitment fraud will require a composite approach that involves deep inspection and analysis of user, organization, and network data.[2]discusses the development of a job prediction system using various deep neural network models.They proposed an ensemble model that combines different deep neural network models, which achieved the highest F1-score of 72.71% in their experiments.

Model/Algorithm/Method

1.Data Collection and Preprocessing:
We collected a dataset of job postings from the kaggle website for the purpose of this project. In the preprocessing phase, several techniques were employed.
Following is the description of the preprocessing technique employed:

- Imputation of missing values in the 'department' column by replacing them with the mode, ensuring representative data.
- Utilization of SimpleImputer for categorical features with a strategy set to "constant" and filling missing values with the label "Missing."
- Application of SimpleImputer for numerical features with a constant fill value, preserving numerical data integrity.
- Implementation of ColumnTransformer to seamlessly integrate imputation strategies, categorizing them into 'data_cat_imp' for categorical features and 'data_num_imp' for numerical features.
- Conversion of the transformed data into a DataFrame for ease of handling and further analysis.
- Separation of features and the target variable ('fraudulent'), with the target variable converted to integer type.
- Use of OneHotEncoder to transform categorical data into a numeric format, facilitating machine learning algorithms.
- Application of ColumnTransformer to selectively apply the OneHotEncoder transformation to categorical features while leaving numerical features unaffected.

- ○ Splitting of the dataset into training and testing sets using the train_test_split function from scikit-learn, ensuring a representative distribution of data for model training and evaluation.

2.Data Visualization:
Inorder to gain insights from the dataset we performed data visualization.

Visualization of the top 10 industries and business functions represented in the dataset.
- ○ The resulting visualization provides a clear representation of the distribution of industries and business functions in the dataset.

Visualization of the distribution of fraudulent job postings in STEM and non-STEM categories.
- ○ The plot effectively illustrates the proportion of real and fraudulent job postings in the STEM category, enhancing understanding at a glance.
- ○ A legend is included to label the segments.

Visualization missing data
- ○ This visualization helps quickly identify the extent and patterns of missing data in the dataset, aiding decisions on how to handle missing values during data preprocessing.

3.Random Forest:
We implemented Random Forest in the following way:

Random Forest Model Training:
- ○ Utilized the RandomForestClassifier from scikit-learn for constructing a Random Forest model.
- ○ The model was trained on the training set ('X_train', 'y_train') using the 'fit' method.
- ○ Subsequently, predictions were made on the test set ('X_test') using the 'predict' method.

Accuracy Measurement:
- ○ The accuracy of the Random Forest model was evaluated using the 'score' method, providing an overall classification accuracy on the test set.
- ○ The accuracy value ('rfc_accuracy') was printed to the console, indicating the percentage of correct predictions.

Hyperparameter Tuning:
- ○ Conducted hyperparameter tuning by varying the number of trees in the Random Forest ensemble.
- ○ Iterated through a set of predefined values for the number of trees ('n_trees'), such as 10, 50, 100, 200, and 300.

Model Evaluation for Different Tree Counts:
- ○ For each value of the number of trees, a new RandomForestClassifier was instantiated and trained on the training set.
- ○ Predictions were made on the test set, and the number of correct predictions was calculated.

- The accuracy, represented as the percentage of correct predictions, was printed for each set of predictions.
- This process provided insights into how the model's performance varied with different tree counts.

4.Naive Bayes:

Naive Bayes Classifier Implementation:

- Utilized the Multinomial Naive Bayes classifier ('MultinomialNB') from the scikit-learn library.

Model Training:

- The Naive Bayes model was trained on the training set ('X_train_dtm'), which represents the document-term matrix derived from the preprocessed text data.
- The training process involved learning the probabilities associated with different terms in the text data.

Prediction on Test Set:

- Employed the trained Naive Bayes model to predict the target variable (fraudulent or not) for the test set ('X_test_dtm').
- The predictions were stored in the variable 'y_pred_nb'.

Model Evaluation:

- The accuracy of the Naive Bayes model was evaluated using the 'accuracy_score' function from the scikit-learn library.
- The accuracy score quantifies the proportion of correctly classified instances in the test set compared to the actual labels ('y_test').

Performance Assessment:

- The accuracy score provides a performance metric, indicating how well the Naive Bayes classifier distinguishes between genuine and fraudulent job postings.
- This metric serves as a key indicator of the model's effectiveness in making accurate predictions.

5.Decision Tree:

Decision Tree Implementation:

Text Vectorization:

- The text data was vectorized using the CountVectorizer, a method that converts a collection of text documents into a matrix of token counts.
- The vectorizer was fitted to the training data ('X_train') and transformed both the training and testing data into document-term matrices ('X_train_dtm' and 'X_test_dtm').

Decision Tree Classifier:

- A Decision Tree Classifier ('DecisionTreeClassifier') was employed for the classification of job postings as either real or fake based on the vectorized text data.
- The model was trained on the document-term matrix of the training set ('X_train_dtm'), learning decision rules from the vectorized text features.

Prediction and Evaluation:

- Predictions were made on the test set ('X_test_dtm'), and the accuracy of the Decision Tree model was evaluated using the 'accuracy_score' function from scikit-learn.
- This accuracy score provides a quantitative measure of the model's performance in classifying job postings.

Example Usage:

- Two example job postings, represented as 'input_text1' and 'input_text2', were used to demonstrate the model's predictive capabilities.
- The text data was transformed into vector format using the previously fitted CountVectorizer.

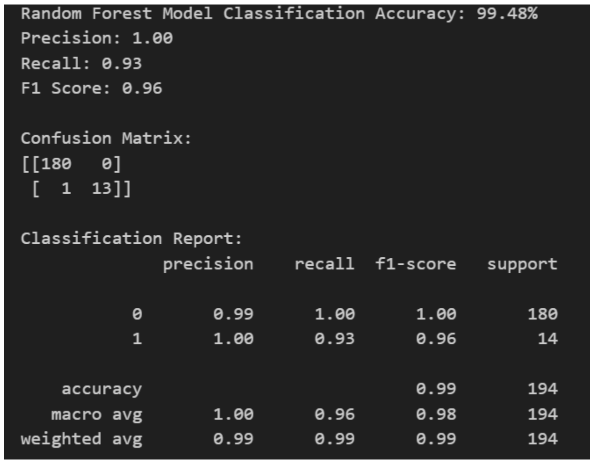
Prediction Display:

- Predictions were made for each example job posting using the trained Decision Tree model.
- The results were displayed, indicating whether each job posting belongs to the real or fake job post category based on the Decision Tree's classification.

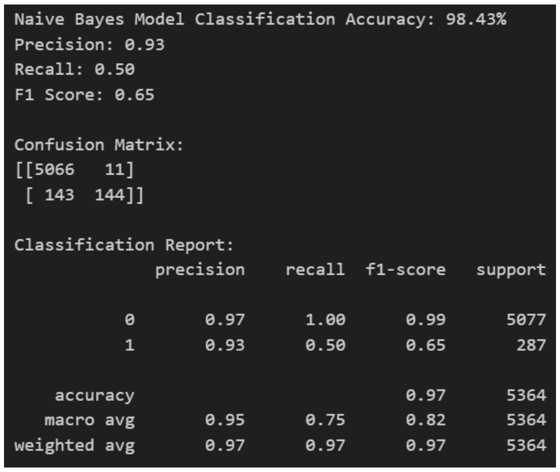
<u>**Results and Findings**</u>

1.Evaluation Metrics:

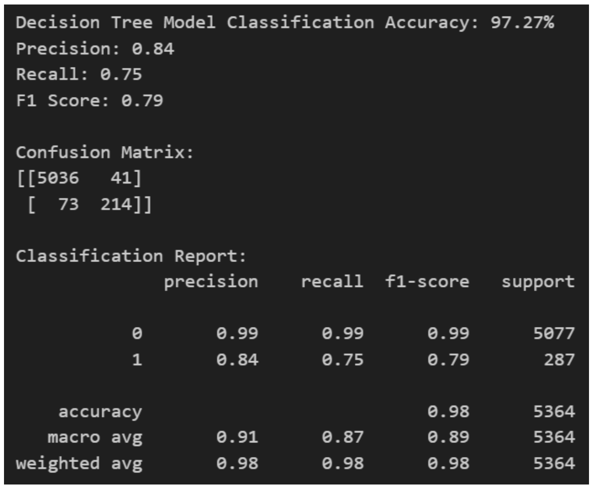
We evaluated the performance of our models using metrics such as accuracy, precision, recall, and F1 score,confusion matrix.



```
Random Forest Model Classification Accuracy: 99.48%
Precision: 1.00
Recall: 0.93
F1 Score: 0.96

Confusion Matrix:
[[180   0]
 [  1  13]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       180
           1       1.00      0.93      0.96        14

    accuracy                           0.99       194
   macro avg       1.00      0.96      0.98       194
weighted avg       0.99      0.99      0.99       194
```



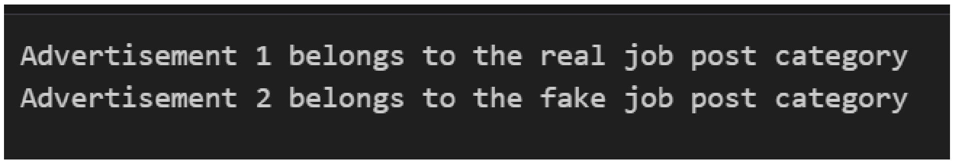
```
Naive Bayes Model Classification Accuracy: 98.43%
Precision: 0.93
Recall: 0.50
F1 Score: 0.65

Confusion Matrix:
[[5066   11]
 [ 143  144]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      5077
           1       0.93      0.50      0.65       287

    accuracy                           0.97      5364
   macro avg       0.95      0.75      0.82      5364
weighted avg       0.97      0.97      0.97      5364
```


```
Decision Tree Model Classification Accuracy: 97.27%
Precision: 0.84
Recall: 0.75
F1 Score: 0.79

Confusion Matrix:
[[5036   41]
 [  73  214]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5077
           1       0.84      0.75      0.79       287

    accuracy                           0.98      5364
   macro avg       0.91      0.87      0.89      5364
weighted avg       0.98      0.98      0.98      5364
```

2.Comparative Analysis:
The Random Forest model outperformed both Naive Bayes and Decision Tree methods, achieving an accuracy of 99.48%, compared to 97.27% for Naive Bayes and 98.43% for Decision Tree. This demonstrates the effectiveness of ensemble learning in addressing the complexities of real/fake job posting detection.

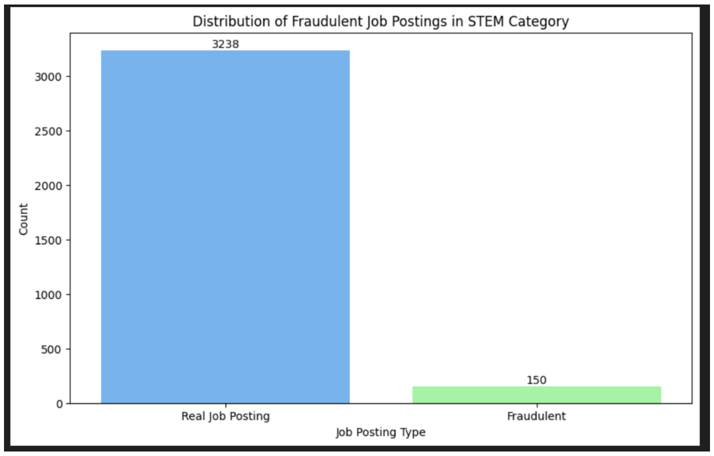
3.Interpretation of Results:
The feature importance analysis revealed that certain keywords, reflecting the content of job postings, emerged as crucial in distinguishing between genuine and deceptive postings.The bar plot visually highlights the top 10 keywords and their importances, offering a concise and accessible representation of the most influential features.The feature importance analysis enhances the interpretability of the Random Forest model, providing insights into the specific elements that contribute significantly to the model's accuracy in predicting real/fake job postings.

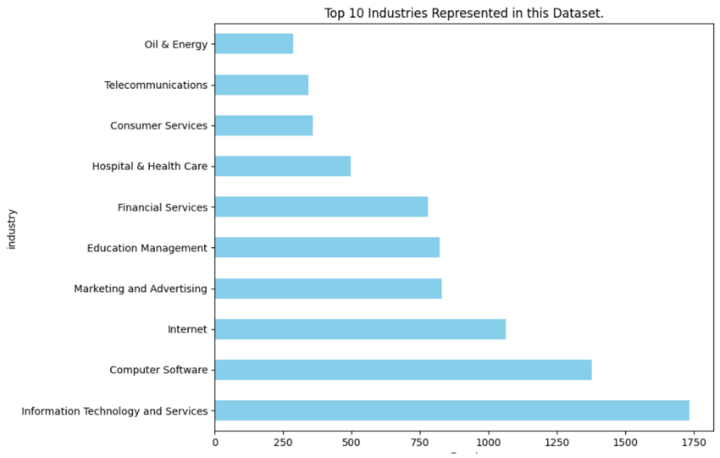
4.Predictions
The predictions offer a practical demonstration of the model's ability to classify job postings as either real or fake.Example advertisements were effectively categorized based on the patterns learned during model training.The following result which we obtained showcases the real-world applicability of the developed models in automatically identifying potentially fraudulent job postings.

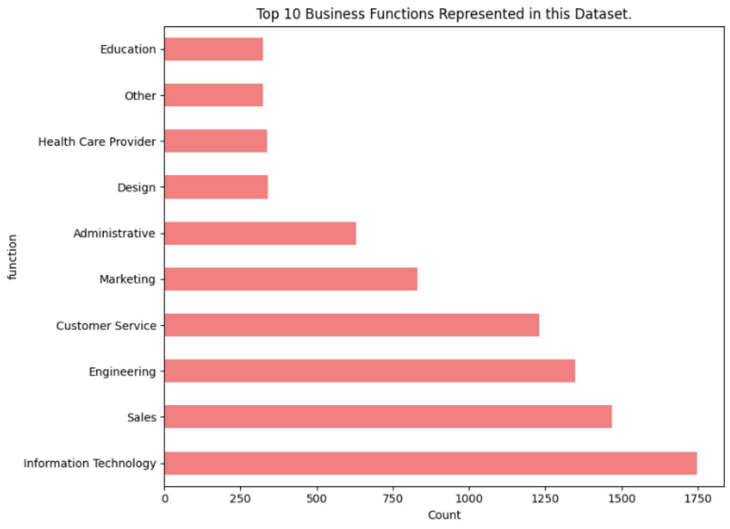

```
Advertisement 1 belongs to the real job post category
Advertisement 2 belongs to the fake job post category
```

5.Visualization

Following are some of the visualizations obtained







Conclusion

In conclusion, our project aimed to tackle the pressing issue of distinguishing between genuine and deceptive job postings on online platforms. The combined use of machine learning algorithms, specifically the Naive Bayes and Decision Tree classifiers and Random Forest, along with effective text preprocessing techniques, has yielded significant insights and outcomes.This project contributes to the ongoing efforts to create a secure and trustworthy environment for job seekers on online platforms. By leveraging machine learning algorithms and text analysis, we provide a robust solution for real/fake job posting prediction, promoting the integrity of the online job market.

References

1. **Automatic Detection of Online Recruitment Frauds: Characteristics, Methods, and a Public Dataset**
2. **Job Prediction: From Deep Neural Network Models to Applications**