Analysis And Classification of Job Postings As Fake / Legitimate

Chetana Chunduru
Department of Computer Science
North Carolina State University Raleigh, NC 27695
cchetan2@ncsu.edu

1. Introduction

In this time of impending recession in the IT sector, the number of cases of fraudulent job posting is increasing. According to the Federal Trade Commission, Americans were scammed out of $68 million in the first quarter of 2022 due to fraudulent business and job opportunities. Employment-related scams have been a persistent problem, but they increased in 2020 as criminals preyed on people who had lost their jobs due to Covid. Employment scams have only become more sophisticated, so it's critical to be cautious as a job seeker. To address this issue, our project aims to leverage the power of Machine Learning models to predict the genuineness of Job postings based on history data.

2. Background

Classification is a technique for categorizing data into a set number of groups. Its main goal is to determine which category/class a new data set belongs to. A classification model attempts to derive some conclusion from the training input values. It will predict the new data class labels/categories. Classification can be used to perform a wide range of tasks, including speech recognition, handwriting recognition, biometric identification, document classification, and so on.

Our project's goal is to be able to classify job postings as legitimate or not based on the data attributes used to train the model. Because we have the output label for the data points, this falls under the Supervised Machine Learning use case, and for our project, we are using classification models such as Decision Tree, K Nearest Neighbours, Bayes Classifier, etc. In addition, we hope to discover valuable patterns in the data by using various visualization techniques to gain a better understanding of the data.

3. Methodology

3.1 Data Selection

The dataset for this project can be found at the link: https://www.kaggle.com/datasets/shivamb/real-or fake-fake-jobposting-prediction. The dataset consists of 17880 data points, of which 866 are classified as fake job postings under the 'fraudulent' attribute. It also consists of data elements such as job title, location of work, department of the job role, salary range and other metadata related to job profile. We are removing Job IDs as they do not contribute to any decision making process. The dataset also contains NULL elements in some of the attributes as well as duplicate entries - for which proper cleaning technique is applied.

3.2 Data Pre-processing

The dataset used in this project has a lot of data points with the values of null, 'Not Applicable' and 'Unsatisfied'. We are replacing those values with the value 'No Info' because all three of these values are corresponding to the same thing. We are removing 282 duplicate rows from the dataset which reduces its size to 17598. Also, the dataset is poorly balanced as most jobs are legitimate, with only a few exceptions.
When we analyzed the accuracy of our models, most of the data points are being classified as genuine or legitimate due to this. So to get over this problem of imbalanced class distribution, we used a technique called SMOTE - Synthetic Minority Oversampling Technique. Instead of just duplicating rows from minority class, there is sampling and k nearest neighbor searching involved and then rows are synthesized with new data. This is important as a well-balanced dataset would yield better results.

3.3 Data Transformation

There are 11 string attributes for which transformation is done to collect valuable information out of it. As the models require the data to be in integer format instead of string format, we converted the categorical fields to integer values by mapping them to an encoding. This ensures that our model would be able to process and train on the input data.

3.4 Data Mining

We are planning to evaluate the below classifiers for our problem statement-

- **K-Nearest Neighbors (KNN)**

  The k-nearest neighbors (KNN) algorithm is a data categorization approach that estimates the chance that a data point will become a member of one group or another based on which group the data points closest to it belong to. The k-nearest neighbor algorithm is a sort of supervised machine learning technique that is used to address classification and regression tasks.

  Things to consider while training a KNN model:
  - Distance metric that is to be used to calculate the distance between two points of data. There are different metrics available, such as Manhattan distance, Euclidean distance, Chebyshev distance or other such metrics.
  - Values of the number of means (k) or the number of groups to be formed
  - Initial seed values for the means

  Reasons to choose this model for our project:
  - Few hyperparameters required for determining for the optimal model
  - Since our data can be non-linear, it would be able to handle it properly as there are no assumptions about the underlying data

- **Decision Tree**

  The Decision Tree algorithm is a member of the supervised learning algorithm family. The decision tree approach, unlike other supervised learning algorithms, may also be utilized to solve regression and classification issues. The purpose of employing a Decision Tree is to build a training model that can predict the class or value of a target variable by learning basic decision rules from training data.

  Things to consider while training a Decision tree:
  - Need to consider the criteria for the split. There are several choices available such as Gini, Entropy etc.
  - Need to handle the missing data in training as well as testing data properly.

- Should consider a max depth of tree parameter to limit the overfitting of the model.

Reasons to choose this model for our project:
- Highly intuitive and easy to understand
- It's a white box model, i.e. very easy to understand the conditions for the decisions of the model ○ Cost of using the model is directly proportional to the depth of the tree and in general cases, the decision making process is faster than other models.

● **Naive Bayes' Classifier**

It is a classification method based on Bayes' Theorem and the assumption of predictor independence. A Naive Bayes classifier, in basic words, asserts that the existence of one feature in a class is independent of the presence of any other feature. The Naive Bayes model is simple to construct and is especially good for very big data sets. In addition to its simplicity, Naive Bayes has been shown to beat even the most advanced classification systems in some use cases.

Things to consider while training a Naive Bayes' model:
- Need to properly stratify data sets to avoid the zero probability phenomena
- If the case arises, apply smoothing techniques if the case of zero probability phenomena is frequent or unavoidable and the results are not as expected.

Reasons to choose this model for our project:
- Simple, fast and easy to implement
- Easy to obtain the estimated probability for a prediction

● **Logistic Regression**

Logistic Regression is a machine learning classification approach. The dependent variable is modeled using a logistic function. The dependent variable is dichotomous in nature, which means that there are only two potential classes (eg.: either the job posting is fraudulent or not). As a result, while working with binary data, this strategy is usually applied.

Things to consider while training a Logistic Regression model:
- Can choose a solver for better convergence of the results based upon the dataset. There are different solvers available for usage such as liblinear, sag, saga, newton-cg. In our implementation, we have used the default solver of 'lbfgs'.
- Can set a penalty strength value (C) for classification of the data while training.

Reasons to choose this model for our project:

- Decision making is fast for classification of unknown records
- The implementation of the model is simple with fewer parameters.

3.5 Interpretation and Evaluation

There are different metrics upon which we can evaluate the performance of our models. Metrics such as Recall / Sensitivity, Precision, F1 Score etc can be utilized. We are specifically focusing on the Sensitivity metrics for evaluation.

The Sensitivity value of a machine learning model is a measure of its ability to detect positive instances. It's also referred to as the true positive rate (TPR) or recall. Sensitivity is used to assess model performance because it shows how many positive instances the model correctly identified. A model with high sensitivity will have few false negatives, implying that it will miss some positive instances. In other words, sensitivity assesses a model's ability to correctly identify positive examples. This is significant because we want our models to be able to detect all positive instances in order to make accurate predictions.

For our project, we have assumed the positive scenario to be a prediction of Fraudulent job posting and a negative to be Legitimate job posting. The scenario of having a False negative, i.e. predicting a job to be genuine whereas in actual it was fraudulent would carry a higher cost as it would have a critical impact on the applicant in terms of monetary status. Since, Sensitivity focuses on reducing the false negatives in the system, we are using this metric for evaluating the models.

4. Experiment Setup

We have implemented the code using the packages such as sklearn for model related functionalities, numpy and pandas for data manipulation, seaborn and matplotlib for data visualization. For collaboration purposes, we have used GitHub, Deepnote and Google Collab so as to work on the project in a distributed manner. For our model evaluation among the different models proposed above, we have utilized the 10-fold cross validation process in our project. We are testing the data using two types of split of the complete data.

Split type 1:
Train 75% - Test 25%

Split type 2:
Train 80% - Test 20%

We then apply the process for model training for each split type, with 10-fold cross validation for each of the models. After getting the cross validation scores based on the recall metric, we are choosing the efficient model out of the 4 available models and then calculating the recall metrics based upon the final test data. A thing to remember is that while training the model, the train data is always preprocessed by SMOTE so as to increase the cases for fake job scenarios and train the model in a better way. We have achieved this by creating a pipeline where we first apply SMOTE on the training data and then pass a model to be trained.

5. Result

After experimenting, we got following test results-

- The data set has 17880 data points, of which 866 are classified as fake job postings under the 'fraudulent' attribute.
- The data set has a total of 70103 null values over all columns.

- There are 282 duplicate rows which are removed from the data set.

- The unique counts for each column in the data set as well as unbalanced data for label 'fraudulent' are visualized.

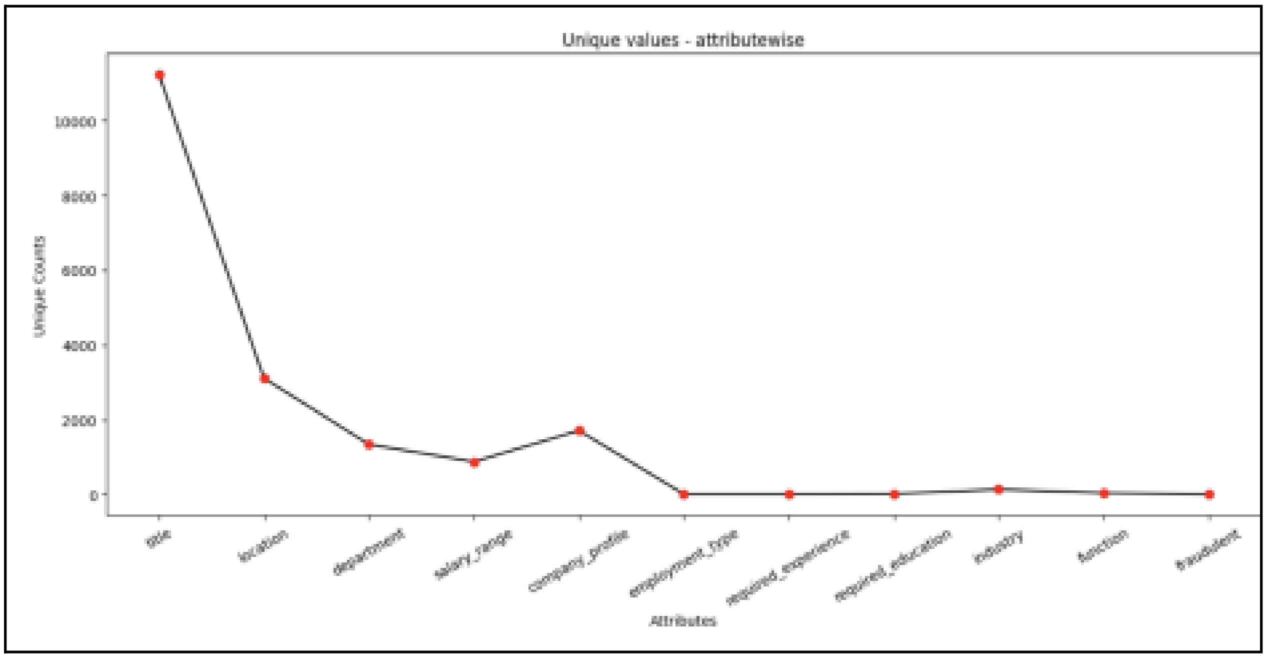


Figure 1: Unique values in the dataset

● The SMOTE Technique increased the count of fake data from the training dataset to 12249 from 649.

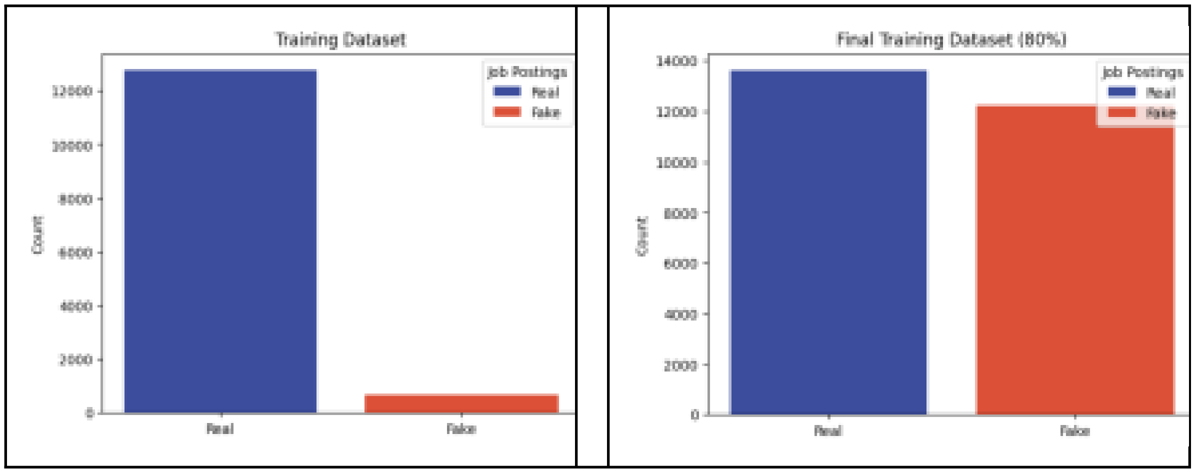


Figure 2: Job posting values in the training dataset before and after applying SMOTE technique

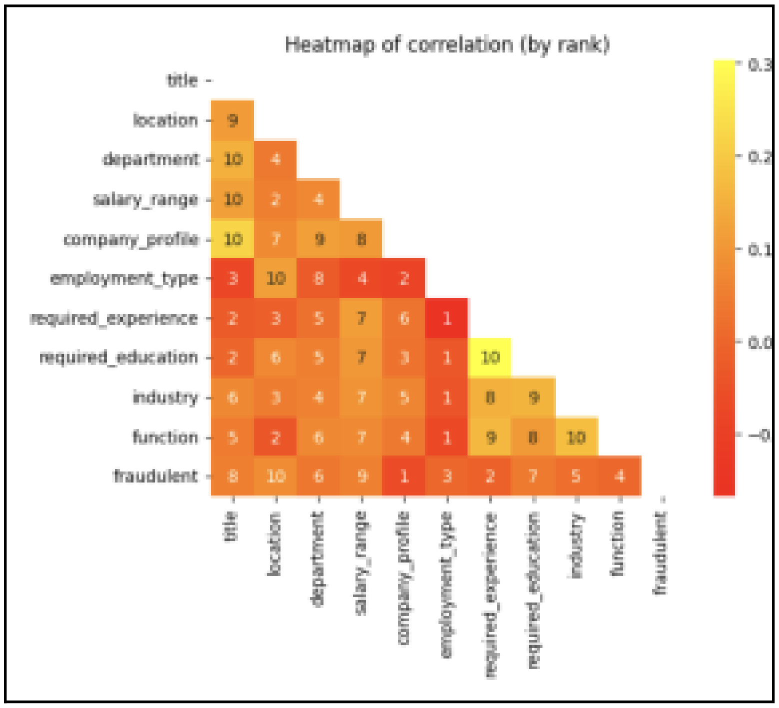
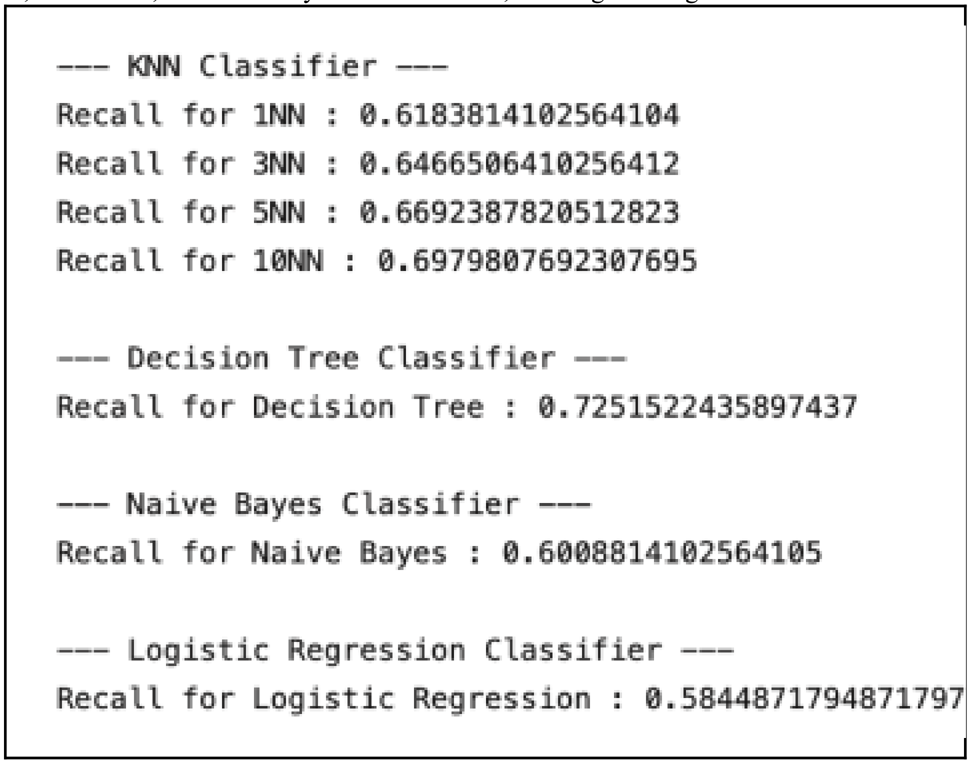


Figure 3: Heatmap of correlations

● Recall of K-Nearest Neighbors for k=10 is ~70% which is better than k=1,3 or 5; for Decision Tree Classifier, it is 72.5%; for Naive Bayes Classifier 60%, and Logistic Regression Classifier is 58.4%.



```
--- KNN Classifier ---
Recall for 1NN : 0.6183814102564104
Recall for 3NN : 0.6466506410256412
Recall for 5NN : 0.6692387820512823
Recall for 10NN : 0.6979807692307695

--- Decision Tree Classifier ---
Recall for Decision Tree : 0.7251522435897437

--- Naive Bayes Classifier ---
Recall for Naive Bayes : 0.6008814102564105

--- Logistic Regression Classifier ---
Recall for Logistic Regression : 0.5844871794871797
```

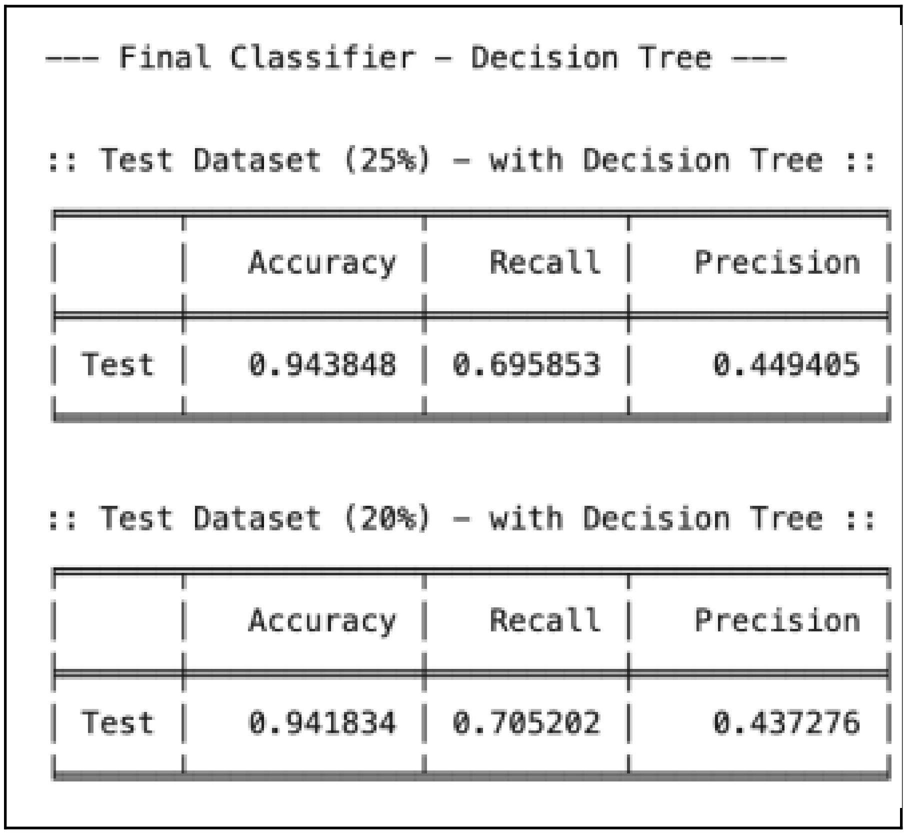
Figure 4: Recall values for KNN, DT, NB and LR Classifiers

6. Conclusion

The real and fake job posting project taken was selected to solve a real world challenging issue. This project aims to provide a solution to this problem by classifying jobs as fake or real job postings. After implementing and observing the results from the 4 models for the recall score, we can conclude that the Decision Tree, although a simple model, performs better than the other models on the cross-validation scores. The KNN model also competes closely with that of the Decision tree, but it lacks the speed in predicting that of the decision tree as it is a lazy evaluation model. Therefore, for our problem statement, we continued with the decision tree model with the final testing data.

Naive Bayes and Logistic Regression model was observed with low values of recall for our problem statement. For Naive Bayes, one major reason for these low scores would be violation of the features independence assumption. There are certain features in the dataset, which logically should depend upon each other. One such example would be Department and fields such as required_education and required_experience. In the real-world data, there are usually such scenarios where some departments prefer people with experience or education, which makes the fields dependent and thus violate the base assumption of Naive Bayes. Similarly, for Logistic Regression, the features should be linearly dependent which was not followed in the current dataset. This would be the reason for the low scores for the Logistic Regression model.

Therefore, we selected the Decision Tree classifier as it was yielding a decent value in comparison to other models for recall. The performance of the model to unseen data was also captured and a recall value of 0.695853 and 0.705202 were observed for different splits for training and testing data.



```
--- Final Classifier - Decision Tree ---

:: Test Dataset (25%) - with Decision Tree ::

╒══════╤════════════╤══════════╤═════════════╕
│      │   Accuracy │   Recall │   Precision │
╞══════╪════════════╪══════════╪═════════════╡
│ Test │   0.943848 │ 0.695853 │    0.449405 │
╘══════╧════════════╧══════════╧═════════════╛

:: Test Dataset (20%) - with Decision Tree ::

╒══════╤════════════╤══════════╤═════════════╕
│      │   Accuracy │   Recall │   Precision │
╞══════╪════════════╪══════════╪═════════════╡
│ Test │   0.941834 │ 0.705202 │    0.437276 │
╘══════╧════════════╧══════════╧═════════════╛
```

Figure 5: Testing on Final Classifier - Decision Tree

References

[1] Github Repository link. “https://github.ncsu.edu/asfirodi/engr-ALDA-Fall2022-P12” [2] Project website.“https://deepnote.com/@alda-space/Fake-job-classifier-project-f9782cd8-5619-46a2-b23e-37504f2 14121”
[3] A.S.M Kayes Syed Mahbub Eric Pardede. “Online Recruitment Fraud Detection: A Study on Contextual Features in Australian Job Industries”.
[4] Sultana Umme Habiba, Md. Khairul Islam,Farzana Tasnim. “A Comparative Study on Fake Job Post Prediction Using Different Data Mining Techniques”.
[5] Kashvi Taunk,Srishti verma,Sanjukta De. “K Nearest Neighbor Algorithm for Learning and Classification”.