PROJECT REPORT

Project Title: Fake Job Recruitment Detection Using Classification

Name: Ethan Rangel-Torres (801342743)

Primary Paper:

- Authors: Dutta, Shawni, and Samir Kumar Bandyopadhyay
- Title: *Fake Job Recruitment Detection Using Machine Learning Approach*
- Year: 2020
- Journal: International Journal of Engineering Trends and Technology (IJETT) – Volume 68 Issue 4
- Link: Fake Job Recruitment Detection Using Machine Learning Approach

1. Introduction

1.1 Problem Statement

Fraud detection plays a critical role in protecting individuals from scams that aim to steal personal or financial information. We encounter fraud detection technologies every day, such as spam filters in email or security alerts in banking apps. In this project, we will focus on specifically detecting fraudulent job postings. Fake job listings can trick applicants into providing sensitive personal information or even paying for fake employment opportunities.

The challenge this project aims to solve is building a system that can accurately identify fraudulent postings to protect job seekers and maintain trust in online job platforms. Successfully addressing this issue allows users to spend more time focusing on real opportunities instead of wasting time and risking harm by interacting with scams.

1.2 Motivation and Challenges

The motivation behind this project stems from personal experiences with fake recruitments and the increasing difficulty faced by job seekers in distinguishing real opportunities from scams. As the job market becomes more competitive, applicants cannot afford to waste time or risk sensitive information.

Some of the key challenges that make fraud detection complex are:

Data Constraints: Most datasets found online contain very few actual fraudulent postings compared to real ones, because there are significantly less fraud postings in the real world. Since having high quality data is important for training models, it is crucial to balance the training set because without it, identifying scams becomes significantly more difficult.

Classification Balance: Misclassifying a scam as legitimate can be dangerous, especially when applicants submit sensitive personal information. At the same time, over predicting fraud can lead to genuine opportunities being unfairly filtered out. Finding the right balance between catching scams and preserving real opportunities is key.

1.3 Summary of Solution

This project develops a Proof of Concept (POC) fraud detection model by testing multiple classification models, applying class balancing techniques, and optimizing model performance through hyperparameter tuning.

The primary research paper for this project, "Fake Job Recruitment Detection Using Machine Learning Approach" by Dutta and Bandyopadhyay (2020), used traditional classifiers such as Decision Tree, Naive Bayes, and ensemble methods like Random Forest. They also used preprocessing steps such as handling missing values, removing stop words, and transforming the dataset into feature vectors, which this project similarly incorporates.

However, instead of following their exact classifier choices, this project explores Logistic Regression, LightGBM, and XGBoost, to evaluate whether more modern approaches could yield better fraud detection results. The goal was to find a model that balances strong precision and recall, helping reduce the risks of both false negatives and false positives.

2. Survey of Literature

1) Amaar, Aashir, et al. Detection of Fake Job Postings by Utilizing Machine Learning and NLP. Neural Processing Letters (Springer), vol. 54

Contribution:
This paper introduced the use of machine learning algorithms combined with TF-IDF and Bag-of-Words feature extraction techniques for detecting fraudulent job postings. Several traditional classifiers were compared, including Random Forest, Logistic Regression, and K-Nearest Neighbors.

Novelty:
- Showed the importance of text feature extraction methods like TF-IDF and BoW for fraud detection.
- Addressed class imbalance using ADASYN oversampling to improve model rare fraudulent samples.

Pros:
- Showed that basic machine learning models can perform well with proper text preprocessing.
- Demonstrates the effectiveness of TF-IDF in improving the classification performance.

Cons:
- Focused mostly on simpler models without hyperparameter tuning for deeper optimization.

- Relied on traditional oversampling rather than exploring alternative balancing methods like SMOTE.

Relevance:
This paper helped implement TF-IDF as the feature extraction method in this project. However, different balancing techniques and model selections were explored, aiming for a more optimized fraud detection solution.

2) Akram, Natasha, et al. Online Recruitment Fraud (ORF) Detection Using Deep Learning Approaches. IEEE Access, vol. 12

Contribution:
This study applied deep learning models such as BERT and RoBERTa to the task of detecting online recruitment fraud. It also explored several strategies to handle class imbalance, including SMOTE and its variations.

Novelty:
- Introduced the application of transformer based deep learning models for fraud detection in recruitment.
- Demonstrated the critical role of dataset balancing, using SMOTE to improve deep learning model performance.

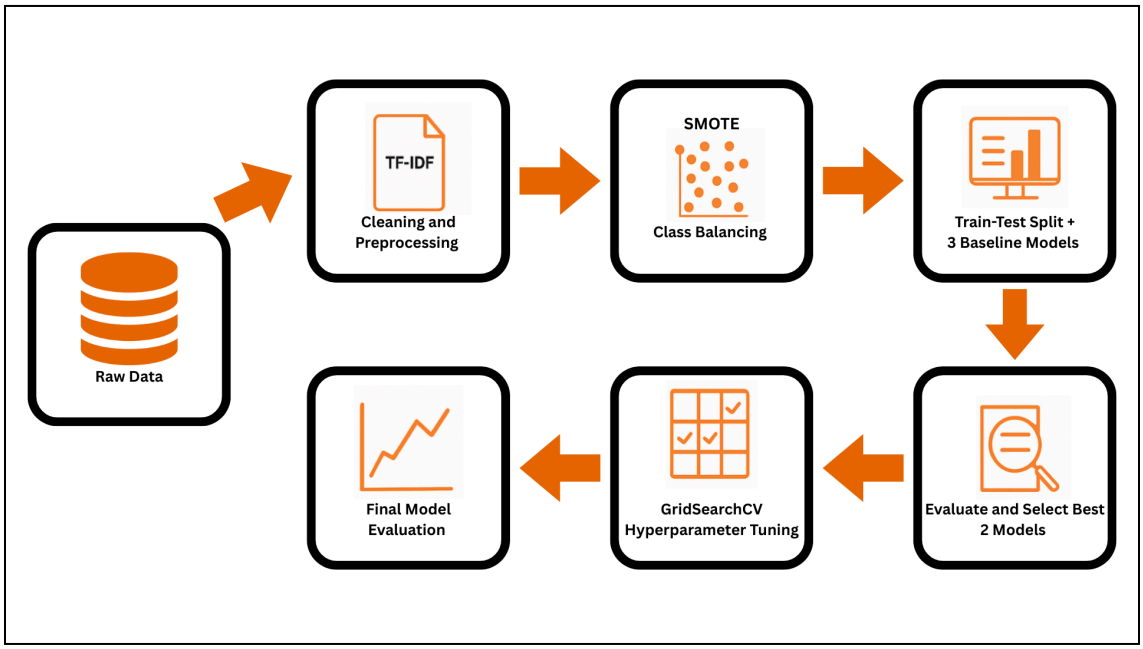
Pros:
- Achieved strong performance by combining data balancing and modern deep learning techniques.
- Provided an overview of evaluations on various resampling methods.

Cons:
- Deep learning models require significantly more computational resources.
- May not be suitable for projects where lightweight and fast solutions are necessary.

Relevance:
This paper motivated the decision to apply SMOTE as a class balancing method in this project. However, instead of transformer models, classic machine learning algorithms were prioritized to keep the solution lightweight and faster for practical use.

3. Methods

3.1 Pipeline Architecture



3.2 Algorithm

1. Data Preparation
- Load the fraudulent job postings dataset.
- Perform cleaning and preprocessing:
  - Combine important text fields (**description and requirements**).
  - Handle missing values by removing them.
  - Select key features (**has_company_logo, has_questions, telecommuting, and text fields**).

2. Feature Extraction
- Apply **TF-IDF Vectorization**:
  - Convert the combined text into numerical feature vectors.
  - Limit to the top 5000 most important words based on frequency.

3. Class Balancing
- Apply **SMOTE (Synthetic Minority Over-sampling Technique)**:
  - Oversample the minority class (fraudulent jobs) to balance the training data.

4. Model Training
- Split the dataset into training and testing sets (**80/20** split).
- Train three baseline models:
  - **Logistic Regression**
  - **LightGBM**
  - **XGBoost**

5. Model Evaluation
- Evaluate each baseline model using:
  - **F1-score**
  - **Precision**
  - **Recall**
  - **Confusion Matrix**
- Select the **top two** performing models based on F1-score and precision.

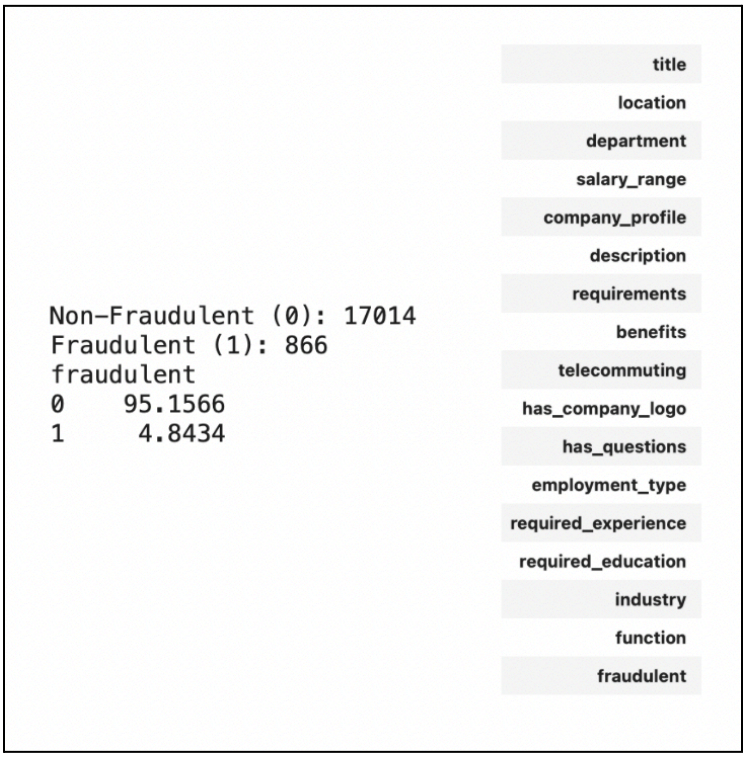
6. Hyperparameter Tuning
- Apply **GridSearchCV** on the selected models:
  - Tune hyperparameters (**num_leaves, max_depth, learning_rate, n_estimators, ect**).
  - Perform cross-validation (**CV=3**) and optimize for **F1-score**.

7. Final Model Evaluation
- Retrain the best-tuned model with optimized parameters.
- Evaluate the final model performance using:
  - **Final F1-score, Precision, Recall**
  - Final Confusion Matrix to visualize **true positives/negatives and errors**

3.3 Key Implementation Details

Dataset:

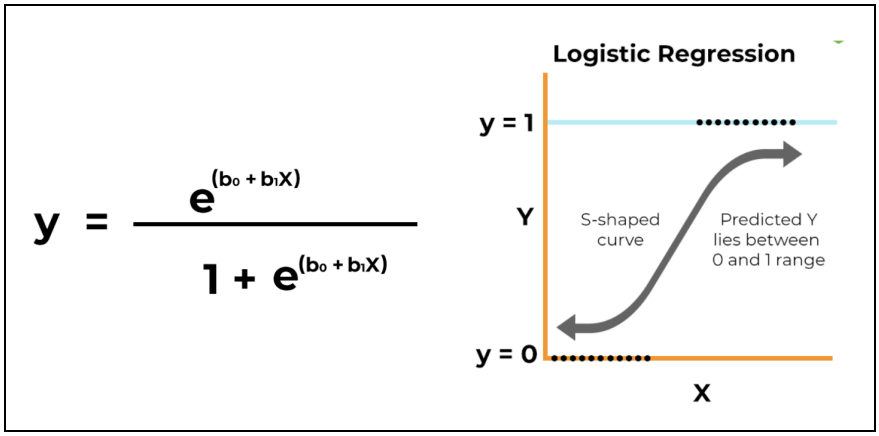
- The dataset has 17 features and 17,880 rows of job postings. Since this is real data, it is significantly unbalanced because there are many more real jobs than fake ones. There is less than 5% of job postings that are actually fraudulent, so in order to train models properly we will have to use class balancing techniques like SMOTE.



*1.The image above was generated using python to display information on the "Real / Fake Job Posting Prediction" dataset from Kaggle. For this project we will train our models on the following features: 'description', 'requirements', 'has_company_logo', 'has_questions', 'telecommuting'***.**

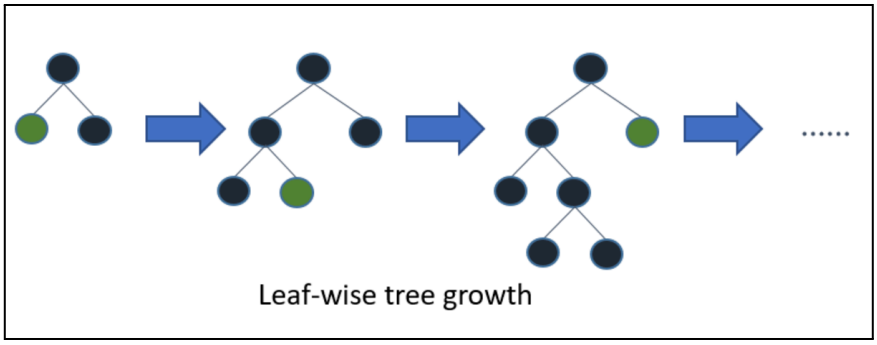
Baseline Models:

- **Logistic Regression:** A supervised linear model that calculates the probability of a binary class (real or fake). It uses a sigmoid function to transform input features into a value between 0 and 1. It's very effective when given linearly separable data and shows interpretable results easily since it's either y = 0 or 1.



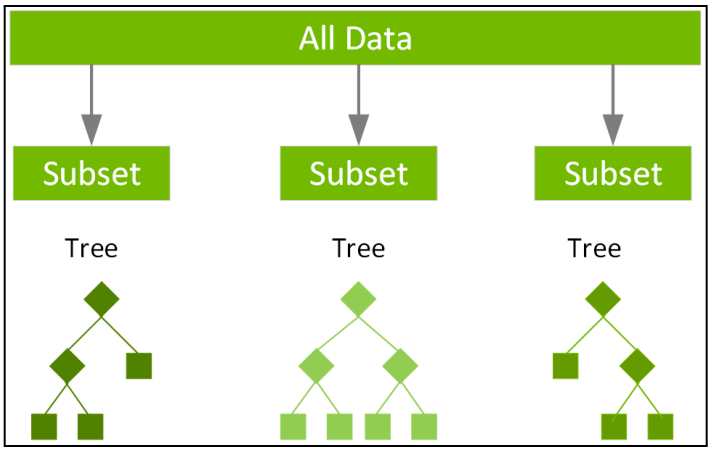
2.The image above displays the sigmoid function used in Logistic Regression to predict the probability of a binary outcome.

- **LightGBM:** A tree based gradient boosting algorithm that is designed to train models on small subsets of data, resulting in faster performance and greater scalability. It is able to improve model training time by using the Leaf-wise tree growth strategy, which is when trees are grown by splitting the leaf with the largest loss reduction first.



3. The image above displays the leaf-wise tree growth strategy used by LightGBM to improve accuracy and speed during model training

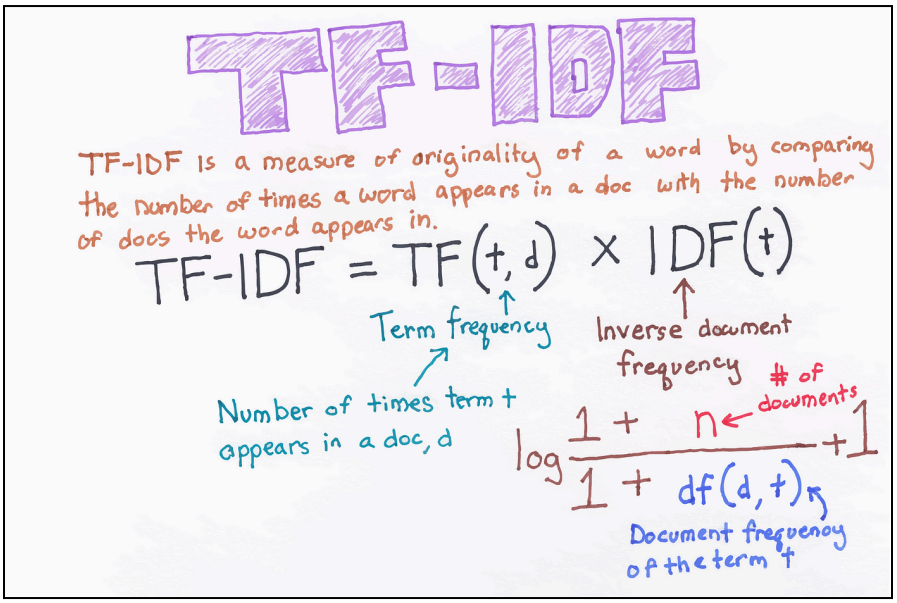
- **XGBoost:** An optimized version of gradient boosting with built in regularization to reduce overfitting while maintaining high predictive power. It is efficient in computation because it divides the dataset into subsets, trains multiple trees in parallel, and combines their predictions to maximize model performance.



4. The image above shows how XGBoost splits the full dataset into multiple subsets and trains decision trees in parallel to optimize model performance.

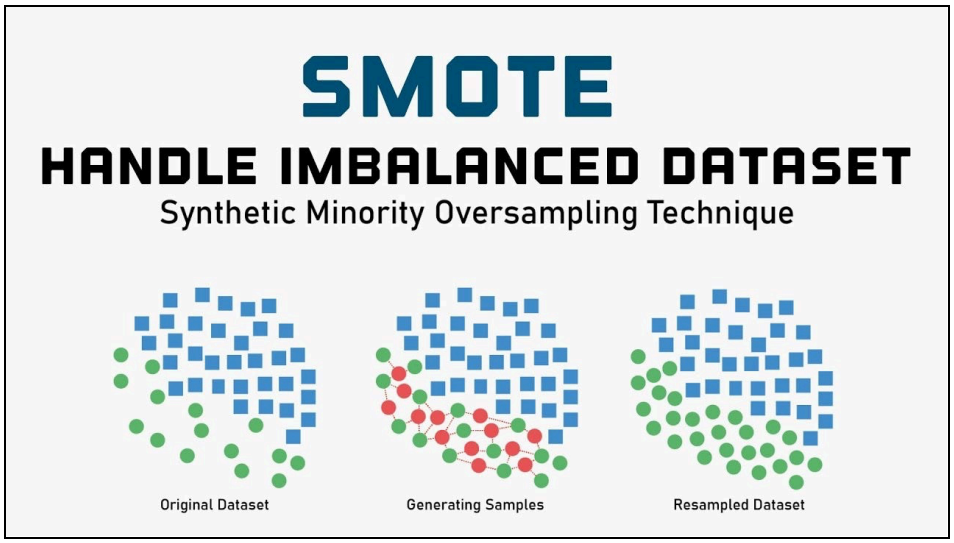
Preprocessing and Class Balancing:

- **TF-IDF (Term Frequency–Inverse Document Frequency):** Converts text data into numerical values by measuring word importance. Reduces weight for common words and boosts rare but informative terms.



5. The image above shows how TF-IDF calculates word importance using term frequency and inverse document frequency.

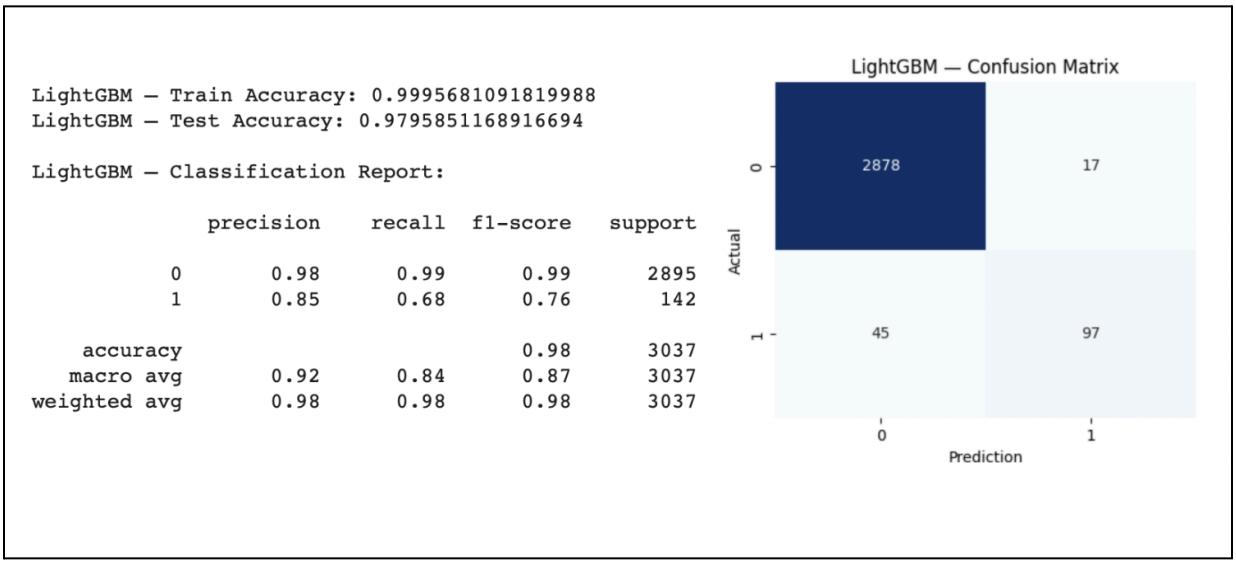
- **SMOTE (Synthetic Minority Oversampling Technique):** Oversamples the minority class by generating synthetic examples to compensate for the class imbalances. It's crucial to generate these synthetic data points for training the model so it doesn't favor the majority (real job postings) class.



6. The image above displays how SMOTE generates synthetic examples to balance the dataset

Evaluation:

- **Metrics:** When evaluating the performance of each model, we focused on several key metrics: Precision, Recall, F1-Score, and Accuracy. Each of these metrics helps capture different aspects of model performance, especially important in fraud detection where catching fraudulent jobs (recall) and minimizing false alarms (precision) are both critical.



7. The image above is an example of the metrics we found from training a baseline LightGBM model. It was made using a python function to display the accuracy, classification report and confusion matrix

4. Experiments

4.1 Replicating Modified Paper Experiments

Experimental Setup

Dataset Preprocessing:

- **Select Features and Splitting Data:** First, missing values were first dropped to ensure clean input. The description and requirements columns were merged to create a new combined_text feature, which consolidated key textual information into a single field. The features selected for model training were ‘combined_text’, ‘has_company_logo’, ‘has_questions’, and ‘telecommuting’. The dataset was then split into training and testing sets using an 80/20 ratio, with stratify=y specified in the split to ensure that the class distribution (real vs. fake jobs) remained balanced across both sets.

```
# Combinding text fields
df['combined_text'] = df['description'] + ' ' + df['requirements']

# Select features to use
features = ['combined_text', 'has_company_logo', 'has_questions', 'telecommuting']
X = df[features]
y = df['fraudulent']

# Splitting data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2, random_state=0)
```

- **Text Vectorization and Class Balancing:** After splitting, the ‘combined_text’ feature was vectorized using TF-IDF to reduce the influence of common words and emphasized more meaningful terms.The other features were then horizontally combined with the text vectors using ‘hstack’, allowing the model to learn from both textual and structured data simultaneously. To fix the significant class imbalance in the dataset, SMOTE was applied to the training data, generating synthetic examples of fraudulent postings and balancing the dataset to improve model performance during training.

```
[ ]  from sklearn.feature_extraction.text import TfidfVectorizer

     # We Implement TF-IDF on 'combined_text'
     tfidf = TfidfVectorizer(stop_words='english', max_features=5000)
     X_train_tf = tfidf.fit_transform(X_train['combined_text'])
     X_test_tf = tfidf.transform(X_test['combined_text'])
```

```
[ ]  from scipy.sparse import hstack

     # extract binary features
     train_bit = X_train[['has_company_logo', 'has_questions', 'telecommuting']].values
     test_bit = X_test[['has_company_logo', 'has_questions', 'telecommuting']].values

     # Use hstack do append the the binary features at end of tfidf features (sparse matrix)
     X_train_combined = hstack([X_train_tf, train_bit])
     X_test_combined = hstack([X_test_tf, test_bit])
```

```
     from imblearn.over_sampling import SMOTE

     # we used SMOTE to generate more fake jobs
     sm = SMOTE(random_state=0)
     X_train_balanced, y_train_balanced = sm.fit_resample(X_train_combined, y_train)
```

Training

- **Training Baseline Model:** Three baseline models were trained (Logistic Regression, LightGBM, and XGBoost). Each model was initialized with a random_state=0 to ensure reproducibility of results. After evaluation, the two models with the strongest performance were selected to move forward for hyperparameter tuning.

```
# Initialize and train model
from lightgbm import LGBMClassifier
lgbm_model = LGBMClassifier(random_state=0)
lgbm_model.fit(X_train_balanced, y_train_balanced)

# Predict
lgbm_preds = lgbm_model.predict(X_test_combined)

# Evaluate
evaluate_model(lgbm_model, X_train_balanced, y_train_balanced, X_test_combined, y_test, "LightGBM")
```

- **Hyperparameter Tuning:** For the best two models, GridSearchCV was applied to improve hyperparameter selection and improve their performance. The models were trained and evaluated using 3 fold cross validation with F1-score as the primary scoring metric. Once the best hyperparameters were identified, the models were retrained and evaluated to measure their final performance.

```
[ ]  from sklearn.model_selection import GridSearchCV
     from lightgbm import LGBMClassifier

     # base model
     lgbm_model = LGBMClassifier(random_state=0)

     # hyperparameters
     lgbm_params = {
         'num_leaves': [15, 31, 63],
         'max_depth': [-1, 10, 20],
         'learning_rate': [0.01, 0.1],
         'n_estimators': [100, 200]
     }

     # Grid Search
     lgbm_grid = GridSearchCV(lgbm_model, lgbm_params, scoring='f1', cv=3)
     lgbm_grid.fit(X_train_balanced, y_train_balanced)
```

```
     # Use best model
     best_lgbm = LGBMClassifier(**lgbm_grid.best_params_, random_state=0)
     best_lgbm.fit(X_train_balanced, y_train_balanced)

     # Evaluate
     evaluate_model(best_lgbm, X_train_balanced, y_train_balanced, X_test_combined, y_test, "Tuned LightGBM")
```

4.2 Results and Discussion

Baseline Models Results:

After training the baseline models and evaluating them on Precision, False Negatives, and F1-Score, the results show the following:

Logistic Regression	LightGBM	XGBoost
Precision: **77%**	Precision: **92%**	Precision: **92%**
False Negatives: **29**	False Negatives: **45**	False Negatives: **45**
F1-Score: **0.65**	F1-Score: **0.76**	F1-Score: **0.76**

These results show that both LightGBM and XGBoost performed identically when using baseline models, and achieved much higher precision and F1-scores compared to Logistic Regression. However, all models initially struggled with minimizing false negatives.

Tuned Models Results:

After tuning the best models with GridSearchCV, the models show the following improvements:

Tuned LightGBM	Tuned XGBoost
Precision: **95%**	Precision: **94%**
False Negatives: **43**	False Negatives: **46**
F1-Score: **0.79**	F1-Score: **0.77**

Both LightGBM and XGBoost improved in precision and F1-score. Tuned LightGBM achieved the best overall performance, with the highest precision and the fewest false negatives, making it the best model for detecting fraudulent job postings from this experiment.

Discussion:

Model Effectiveness: LightGBM and XGBoost both showed they were strong candidates for detecting fraudulent job postings. Even before tuning, they outperformed Logistic Regression in both precision and F1-score. After hyperparameter tuning, LightGBM achieved the highest overall performance, combining strong precision with a lower number of false negatives. This is particularly important in fraud detection, where missing fraudulent cases can cause significant harm to users.

Practical Relevance: The results suggest that LightGBM if further tuned and tested could be realistically deployed on job platforms to flag suspicious listings. By minimizing

false negatives and improving precision, the model would not only catch more scams but also avoid discouraging real job seekers by incorrectly flagging legitimate postings.

Challenges: The models were able to improve a small amount with hyperparameter tuning but even the best model still had 43 false negatives. In fraud detection, even a few missed scams can negatively impact user trust, so further improvements like picking more hyperparameters to test with GridSearchCV will be necessary before using this model in the real world.

5. Conclusion

This project successfully developed a POC fraud detection system using a combination of TF-IDF, SMOTE, and classic machine learning classifiers. After training and evaluation, LightGBM was crowned as the strongest performer, achieving a precision of 95% and an F1-score of 0.79 after hyperparameter tuning.

While the performance metrics did not quite match the higher scores achieved by the models in the primary paper I referenced, this outcome was expected. I intentionally explored models such as Logistic Regression, LightGBM, and XGBoost to investigate simpler but widely used alternatives that were not used in the paper. This decision allows us to understand how traditional methods could still offer competitive results when paired with solid preprocessing and balancing techniques.

Some important lessons learned from this project were the importance of handling class imbalance, feature engineering, and the significant impact of hyperparameter tuning through tools like GridSearchCV. Challenges such as the high number of false negatives highlighted areas for further work and optimization. In the future, I can try to make a model that focuses on multi classification. This would allow us to classify some posting as “sketchy” or a “warning” category if we are not sure if they are fraudulent or real.

Overall, this project demonstrated that even with simpler models and a streamlined pipeline, it is possible to build a high performing fraud detection system capable of being implemented in real world applications on job listing platforms.

6. My Contributions

Previous Work:

The primary foundation for this project was based on the paper:

Shawni Dutta and Prof. Samir Kumar Bandyopadhyay. Fake Job Recruitment Detection Using Machine Learning Approach. International Journal of Engineering Trends and Technology (IJETT), Volume 68, Issue 4, April 2020

The original work used the Kaggle “Real or Fake Fake Job Posting Prediction” dataset and applied machine learning classifiers like Decision Trees, Naive Bayes, K-Nearest Neighbors, and ensemble models like Random Forest and Gradient Boosting. Their preprocessing steps included missing value removal, attribute elimination, and preparing the data for model fitting using categorical encoding.

Link to Paper

My Contributions:

Implementation:

While the dataset and general preprocessing inspiration were drawn from the primary paper, I chose to explore a different set of models that were not the main focus of the original research. I applied Logistic Regression, LightGBM, and XGBoost to evaluate their effectiveness in fraud detection. I also incorporated TF-IDF vectorization for handling the text-based features and SMOTE oversampling to address the issue of class imbalance.

Experimentation:

I designed a pipeline where 3 baseline models were first trained with default parameters. The top two models were then fine tuned using GridSearchCV to optimize results. Evaluation prioritized Macro Precision, F1-Score, and False Negatives to carefully balance fraud detection without losing real opportunities

7. References

- Dutta, Shawni, and Samir Kumar Bandyopadhyay. "Fake Job Recruitment Detection Using Machine Learning Approach." International Journal of Engineering Trends and Technology (IJETT), vol. 68, no. 4, Apr. 2020.
  Link to Paper
- Amaar, Aashir, et al. "Detection of Fake Job Postings by Utilizing Machine Learning and NLP." Neural Processing Letters (Springer), vol. 54, 2022.
  DOI Link
- Akram, Natasha, et al. "Online Recruitment Fraud (ORF) Detection Using Deep Learning Approaches." IEEE Access, vol. 12, 2024.
  DOI LINK
- Spiceworks. (2023). What is Logistic Regression? Spiceworks Tech.
  Link to Article (used image)
- Mandot, P. (2020). What is LightGBM, how to implement it, and how to fine-tune its parameters. Medium.
  Link to Article (used image)
- NVIDIA. (n.d.). What is XGBoost? NVIDIA Glossary.
  Link to Article (used image)
- Shivam Bansal. (2019). Real or Fake? Fake Job Posting Prediction Dataset. Kaggle.
  Link to Dataset
- Analytics Vidhya. (2019). Demonstrating Calculation of TF-IDF from sklearn. Medium.
  Link to Article (used image)
- Python Plain English. (2023). Balancing the Scales: How SMOTE Transforms Machine Learning with Imbalanced Data. Medium.
- Link to Article (used image)
- Link to Github Repo

(Anonymous) Sharing agreement:

- Do you agree to share your work as an example for next semester? **Yes**
- Do you want to hide your name/team if you agree? **Yes**