Foundation of Data Science: Project Report

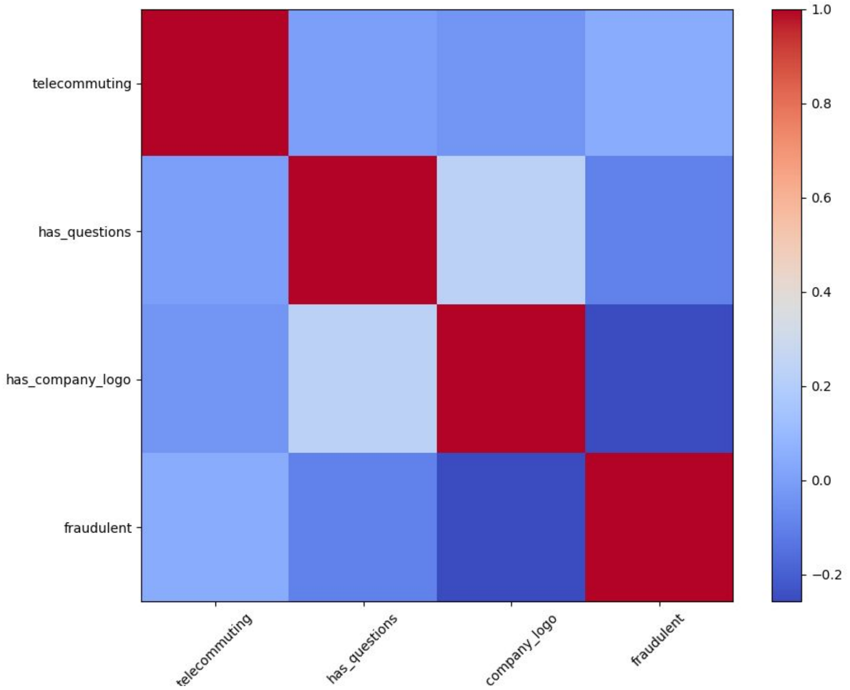
Rahul Kenneth Fernandes

Fraudulent Job Posting Detection

- Objective:
    - To build a machine learning classifier to detect fraudulent job postings based on their title, description and meta data.
    - To expand on existing base models to improve F1 score for class:
      fraudulent == 1

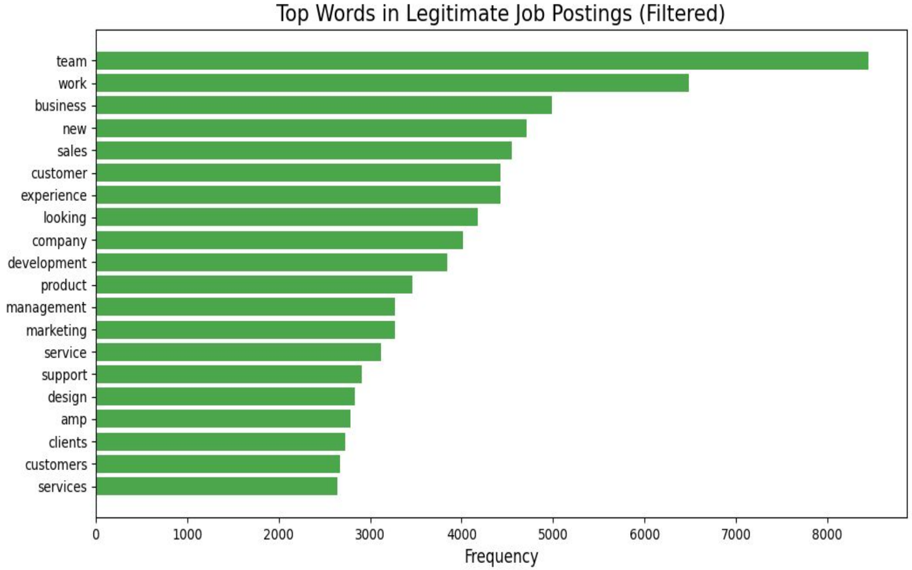
Exploratory Data Analysis

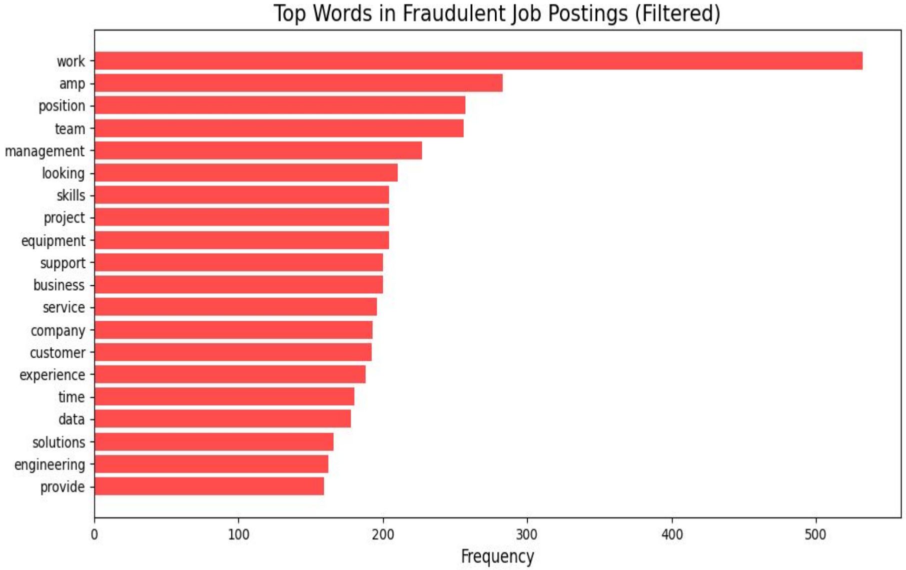
- Imbalance: fraudulent samples consist of 5.1% of the dataset
- Missing data:
  - ‘requirements’: 14.8%
  - ‘location’: 1.7%
  - ‘description’: 0.01%
- Correlation matrix:
  - ‘telecommuting’: very low
  - ‘has_questions’: low
  - ‘has_company_logo’: moderate



Exploratory Data Analysis

Common words used in legitimate and fraudulent postings:





Preprocessing

Data Cleaning:

- Missing data is replaced with an empty string: “”
- Irrelevant columns dropped: ‘requirements’, ‘location’, ‘telecommuting’
- The description column contains most valuable information to predict the authenticity of the posting

```python
# Replace missing values with empty strings
data = data.fillna("")
```

```python
def _preprocessing(self, X, run_type):

    X_set = X.drop(['telecommuting', 'requirements', 'location'], axis=1)

    if (X_set.isnull().any().sum() > 0):
        print('There are null values!!')
        print("Percentage of null values per column:")
        print(X_set.isnull().sum() / len(X_set) * 100)
```

Preprocessing

Transformation & Feature Engineering:

1. Engineered features: Length of characters in the description, number of capitalized words, unique word ratio
2. Combined ‘title’ + ‘description’: converted to lowercase, removed punctuation, numbers and stop words
3. Text Featurization: TF-IDF is used to convert the text data into a sparse matrix with 1 vector for each record (document)
4. Combining all the features:
    a. Numerical features -> StandardScaler()
    b. TF-IDF, binary and numerical features are combined and converted to a dense matrix

Model Training

1. SMOTE Oversampling:
   a. Number of samples = majority - minority
   b. Number of neighbors = 7
2. Voting Classifier with soft voting:
   a. MLPClassifier (Multi-Layer Perceptron)
   b. LogisticRegression
   c. RandomForestClassifier

```
self.ensemble_models = {
    'MLPClassifier': MLPClassifier(
        hidden_layer_sizes = (50,50),
        alpha = 0.0001,
        max_iter = 500,
        activation ='relu',
        random_state = 42,
        verbose = True
    ),
    'RandomForestClassifier': RandomForestClassifier(
        max_depth = 40,
        n_estimators = 100,
        random_state = 42,
        verbose = True
    ),
    'LogisticRegression': LogisticRegression(
        C = 100,
        solver = 'liblinear',
        max_iter = 100,
        random_state = 42,
        verbose = True
    )
}
```

Model Training

- Why this combination?
  - Diversity: Each model approaches the problem differently
  - Complementary strengths:
    - LogisticRegression (Linear): provides a solid baseline
    - MLPClassifier (Neural Network): captures non-linear relationships
    - RandomForestClassifier (Ensemble): handles non-linearity and feature interactions
- Soft Voting:
  - Averages the predicted probabilities from all classifiers and selects the class with highest average probability
  - Balances out biases of individual models

Prediction

- Unseen data is preprocessed using the same _preprocessing function
- The trained model is used to make predictions on a holdout set.

```python
def predict(self, X):

    X_processed = self._preprocessing(X, run_type='test')
    print(f"Original X shape: {X.shape}")
    print(f"Processed X shape: {X_processed.shape}")

    predictions = self.model.predict(X_processed)
    return predictions
```

Model Evaluation

- The best performing classifier model achieved a F1 score of 0.82 on the holdout test set
- Run time: 3.06 minutes

Classification report for best model:

```
Classification Report (Test Set):
              precision    recall  f1-score   support

           0     0.9873    0.9982    0.9927      1709
           1     0.9500    0.7215    0.8201        79

    accuracy                         0.9860      1788
   macro avg     0.9686    0.8599    0.9064      1788
weighted avg     0.9856    0.9860    0.9851      1788
```

F1 score (fraudulent==1) stats for 25 runs:

```
Mean = 0.7972743505277192
Median = 0.7848101265822786
Min = 0.7549668874172185
Max = 0.8201438848920861
STD = 0.02152406075728124
Var = 0.0004632851914831343
Below 10th percentile = 0.7709660789772141
Below 25th percentile = 0.7848101265822786
prob_below_0_75 for MLP, LR & RF = 0.014033484507954589
```

Thank You!