Predicting Real or Fake Job Posting Using Machine Learning

By Buvana
Mini Project 3

TABLE OF CONTENTS

Problem Statement

Predict the **probabilities** of fake job postings.

Objective

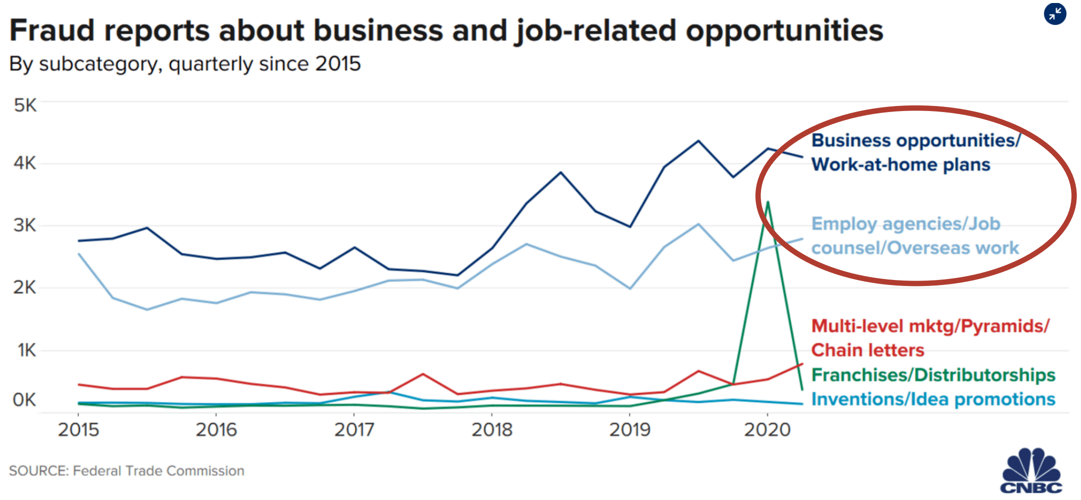
To build machine learning models using data to predict which **job postings are real or fake** and to draw insightful data.

Scope

Use of models to perform **text data classification and to deploy models on the features selected from forward feature selection**.
Use of accuracy metrics to evaluate the model results

Context

- Fake job advertisements found in recruitment portals –can go detected or undetected.
- Exponential increase in fraudulent job opportunities
- In USA- Spike in fraud job adverts/ spams in Q2 of 2020 (Source: CNBC)
- **For jobseekers: Waste of time and effort applying for non existent jobs**
- **For companies/recruiters : Reputation and credibility**



Data

benefits	telecommuting	has_company_logo	has_questions	employment_type	required_experience	required_education	industry	function	fraudulent
NaN	0	1	0	Other	Internship	NaN	NaN	Marketing	0
What you will get from usThrough being part of...	0	1	0	Full-time	Not Applicable	NaN	Marketing and Advertising	Customer Service	0
NaN	0	1	0	NaN	NaN	NaN	NaN	NaN	0
Our culture is anything but corporate—we have ...	0	1	0	Full-time	Mid-Senior level	Bachelor's Degree	Computer Software	Sales	0
Full Benefits Offered	0	1	1	Full-time	Mid-Senior level	Bachelor's Degree	Hospital & Health Care	Health Care Provider	0
NaN	0	0	0	NaN	NaN	NaN	NaN	NaN	0
Your Benefits: Being part of a fast-growing co...	0	1	1	Full-time	Mid-Senior level	Master's Degree	Online Media	Management	0
Competitive Pay. You'll be able to eat steak e...	0	1	1	NaN	NaN	NaN	NaN	NaN	0
NaN	0	1	1	Full-time	Associate	NaN	Information Technology and Services	NaN	0
NaN	0	1	0	Part-time	Entry level	High School or equivalent	Financial Services	Customer Service	0

- ❖ 17880 Rows
- ❖ 17 Columns

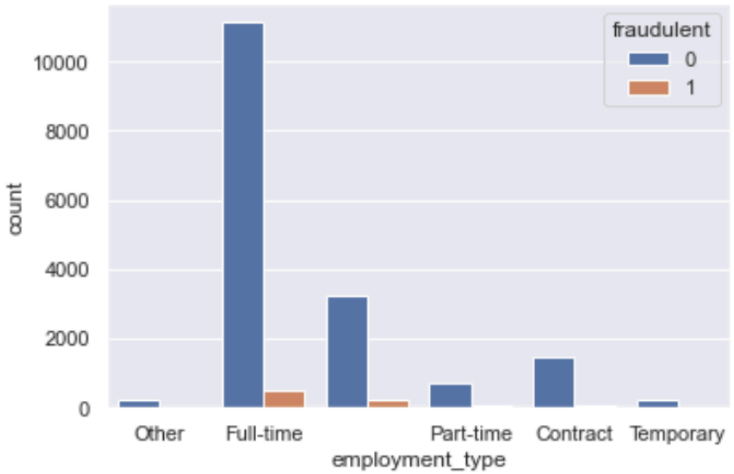
- ❖ Text Variables

1. Title
2. Company profile
3. Description
4. Requirements
5. Benefits

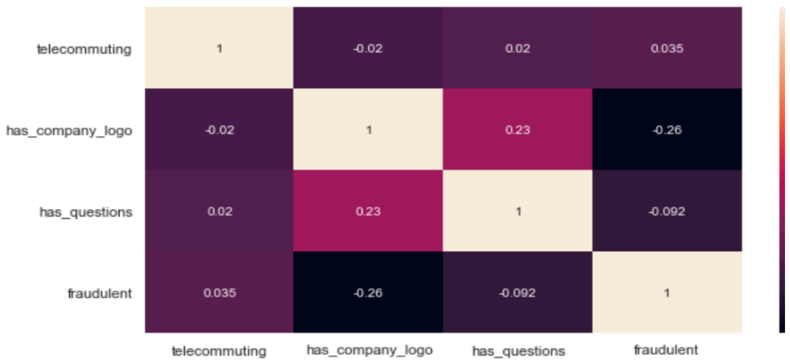
- ❖ Target Variable

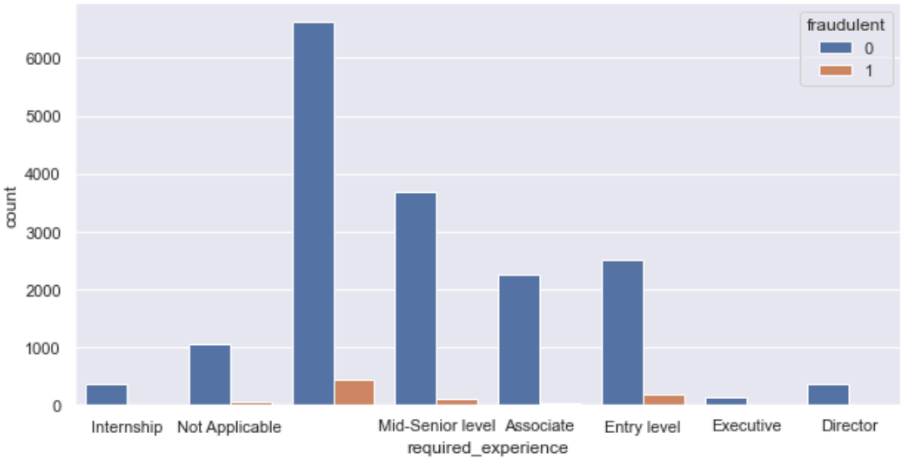
Fraudulent

EDA

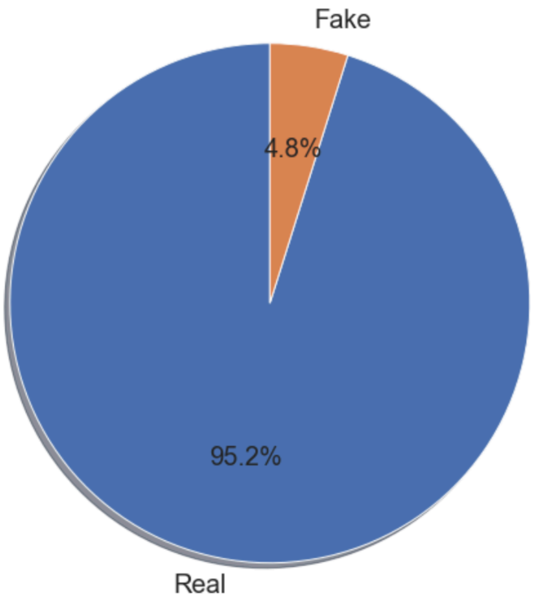


Correlation Matrix





EDA



- Unbalanced data points- Only 4.8% of job postings are fake
- **Resampling method** to balance out the minority class.
- Initially 4.8% of the fraudulent columns denoted fake postings. With resampling, we inflated the count to reflect 50% -50% of real and fake job listings

Feature Extraction

- Basic feature extraction to the 5 Text Columns to check for any distinct features that could differentiate a real and a fake job posting
- **Word Count**
- **Character Count**
- **Word Density**

char_count_d	char_count_r	char_count_b	char_count_t	char_count_cp	word_density_d	word_density_cp	word_density_r	word_density_t	word_density_b
752	715	0	15	710	6.064516	5.035461	6.217391	7.500000	0.000000
1648	1172	1022	35	909	5.333333	6.060000	6.267380	5.833333	4.542222
299	1173	0	34	728	5.980000	5.352941	7.152439	8.500000	0.000000
2213	1228	669	28	509	6.414493	5.988235	7.057471	5.600000	6.968750
1300	653	19	17	1327	7.142857	6.473171	7.337079	5.666667	6.333333

Text Cleaning

Cleaning of text variables

- Removal of Stop words
- Lemmatized words
- Tokenization

Feature Engineering

Count Vectorisation

- Each of the 5 text columns were fit with Count Vectorizer one by one
- Transformed each text column and concatenated the data frames of 5 text columns to perform the modelling

Text Classification

Model analysis- Text columns as the predictor columns

- **Support Vector**
- **Logistic Regression**
- **Naïve Bayes Classifier**
- **Boosting (Gradient Descent)**

	Desc	Desc+Title	Desc+Title+Req	Desc+Title+Req+cp	Desc+Title+Req+cp+Benefits
SVM	0.504195	0.494687	0.500839	0.494966	0.508110
Logreg	0.493568	0.494407	0.501398	0.496085	0.493009
Naive Bayes	0.494128	0.501957	0.497763	0.515660	0.510347
Gradient Boost	0.504474	0.496085	0.493009	0.501957	0.501398

- The highest accuracy level is only 51.6% when we run Naïve Bayes on all the 4 text columns of description, job title, requirements, company profile.

Modelling -Feature Selection

Forward feature selection resulted:

- **Predictor columns (X)** of : *'telecommuting', 'has_company_logo', 'has_questions','required_experience', 'required_education', 'industry', 'function','word_count_t', 'word_count_r','char_count_b','word_density_cp', 'word_density_r', 'description1', 'title1','requirements1', 'benefits1', 'company_profile1'* (from 34 to 18 variables)
- **Target Variable (Y):** *'fraudulent'*

Modelling -Feature Selection

Machine Learning Models were rerun

- Support Vector
- Logistic Regression
- Naïve Bayes Classifier
- Boosting (Gradient Descent)

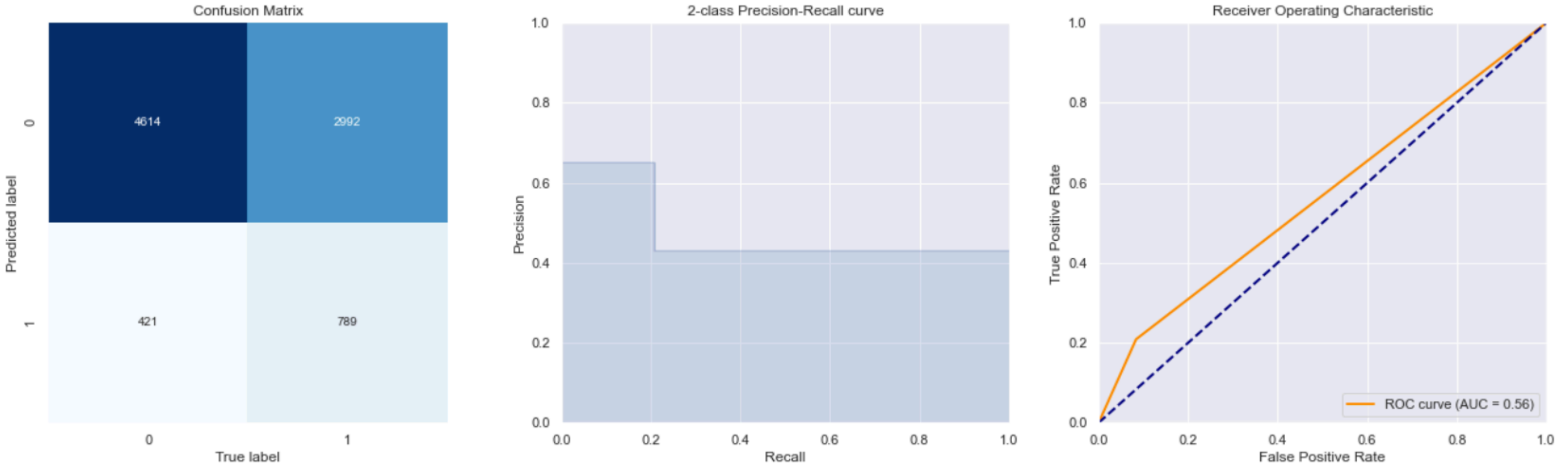
With…

- Train-80% Split
- Test-20% Split
- Cross Validation- 10 folds

Support vector

<u>Training Set</u>

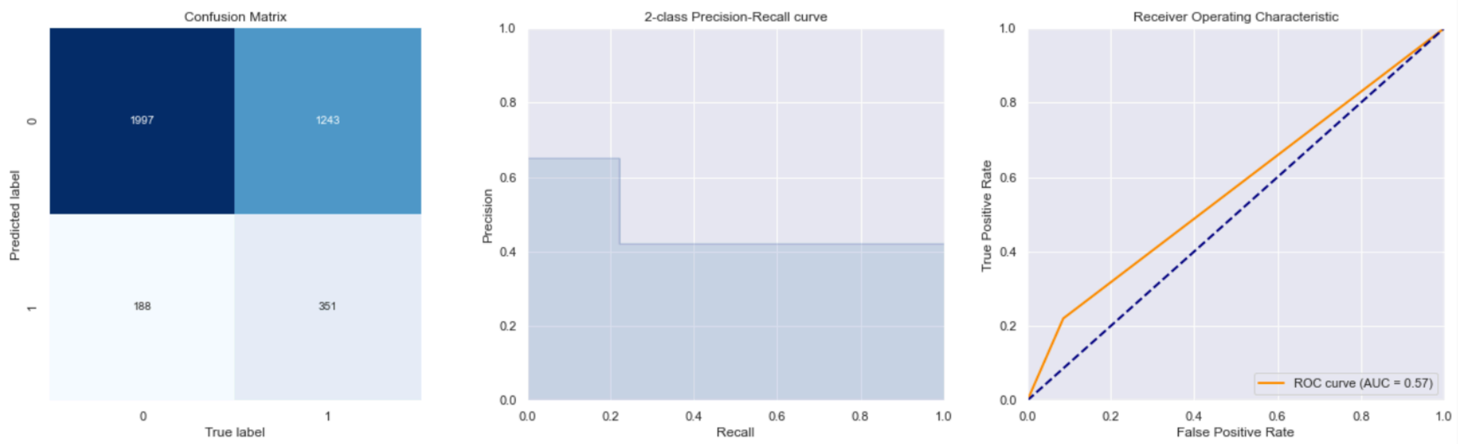
```
********
* SVM *
********
Accuracy : 0.6129 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.6521 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.2087 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.5625                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Support vector

Test Set

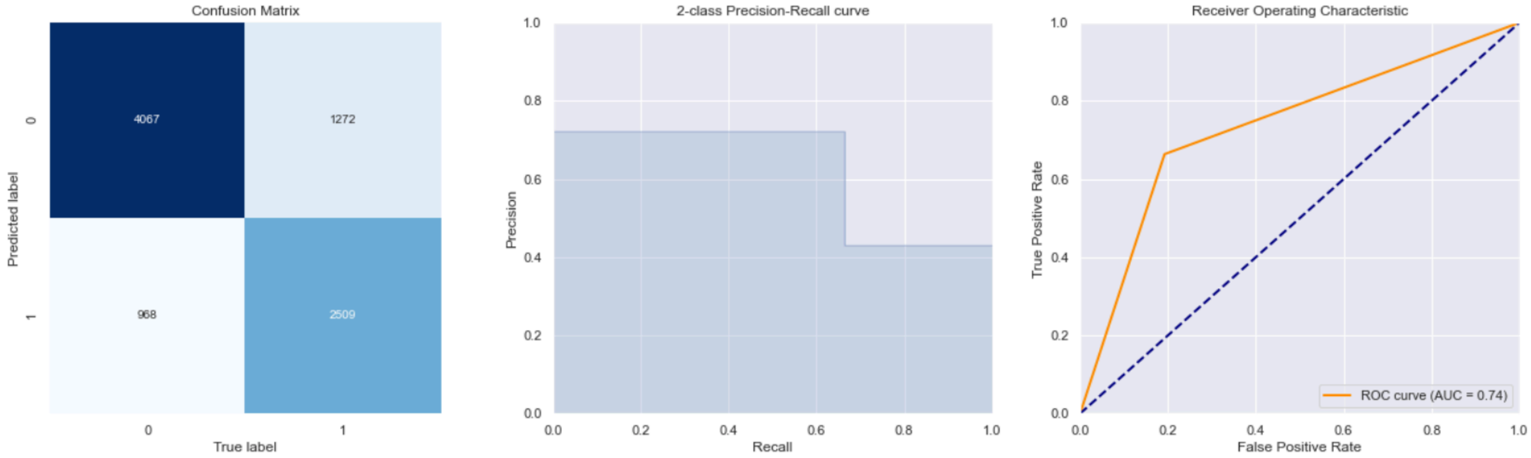
```
Accuracy : 0.6213 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.6512 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.2202 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.5671                                                                     Best: 1, Worst: < 0.5
-------------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Logistic

Training Set

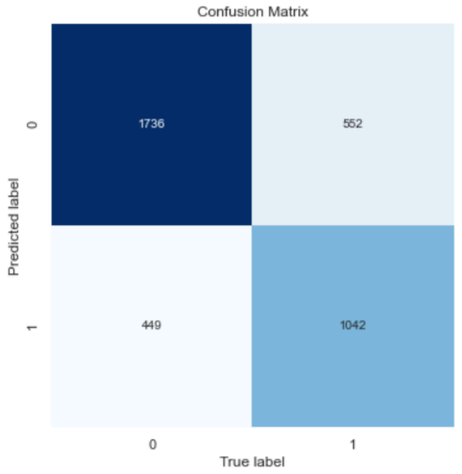
```
*************
* Logistic *
*************
Accuracy : 0.7459 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.7216 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.6636 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.7357                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```

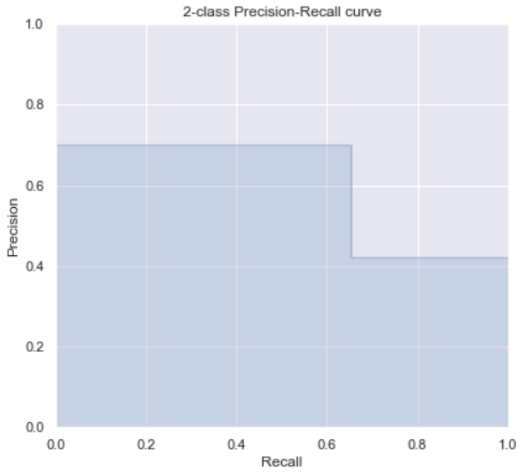


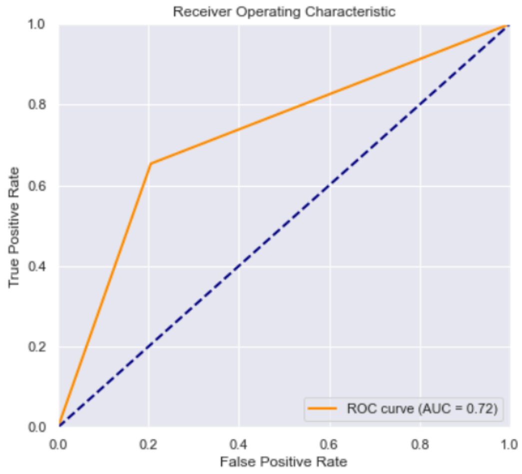
Logistic

Test Set

```
Accuracy : 0.7351 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.6989 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.6537 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.7241                                                                     Best: 1, Worst: < 0.5
--------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



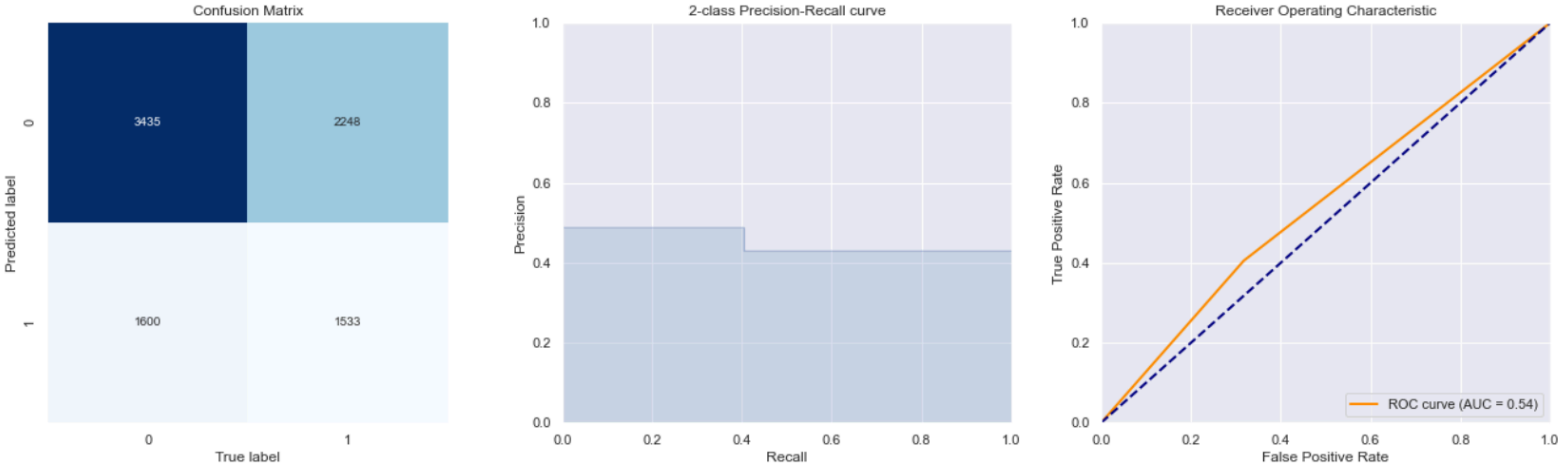




Naïve Bayes

Training Set

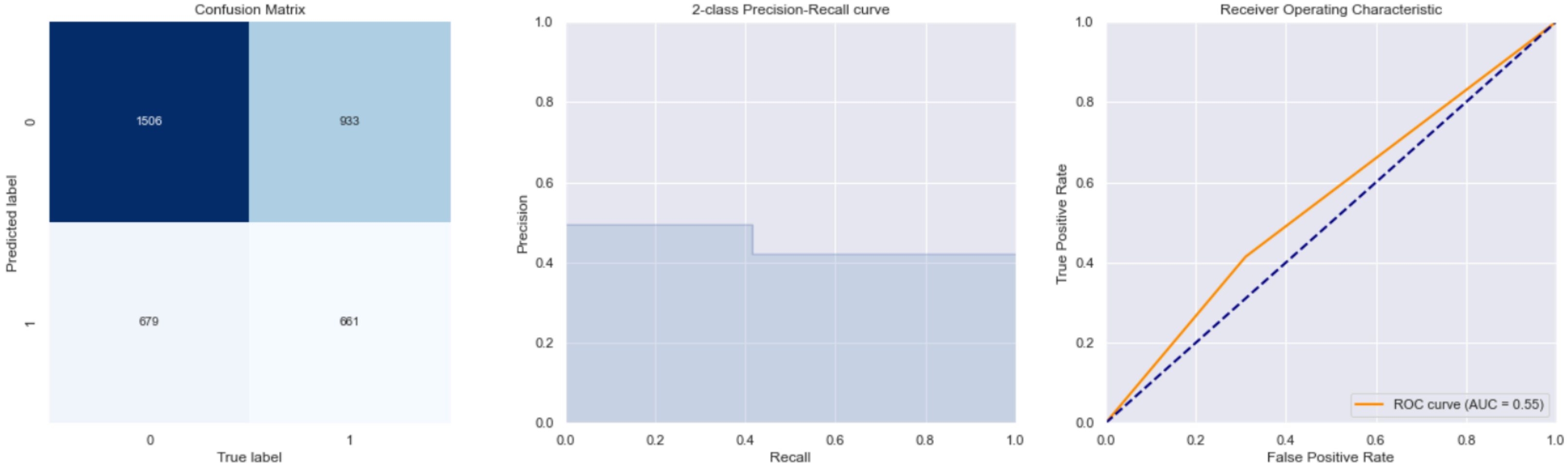
```
****************
* Naive Bayes *
****************
Accuracy : 0.5635 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.4893 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.4054 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.5438                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



Naive Bayes

Test Set

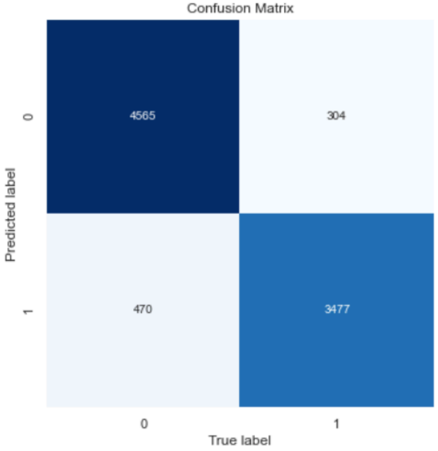
```
Accuracy : 0.5734 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.4933 [TP / (TP + FP)] Not to label a negative sample as positive.       Best: 1, Worst: 0
Recall   : 0.4147 [TP / (TP + FN)] Find all the positive samples.                    Best: 1, Worst: 0
ROC AUC  : 0.5520                                                                    Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```

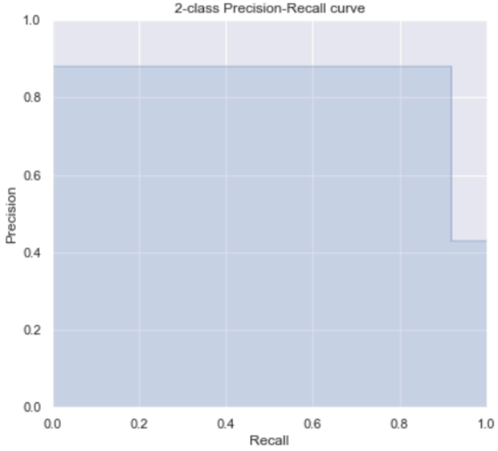


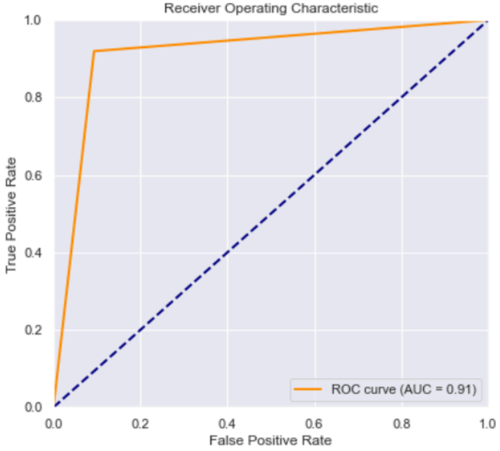
Boosting

Training Set

```
*************
* Boosting *
*************
Accuracy : 0.9122 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8809 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.9196 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.9131                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```



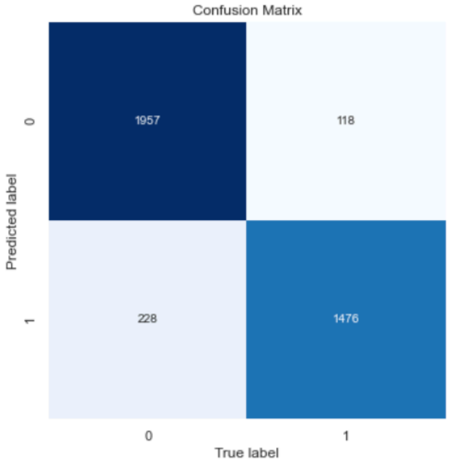


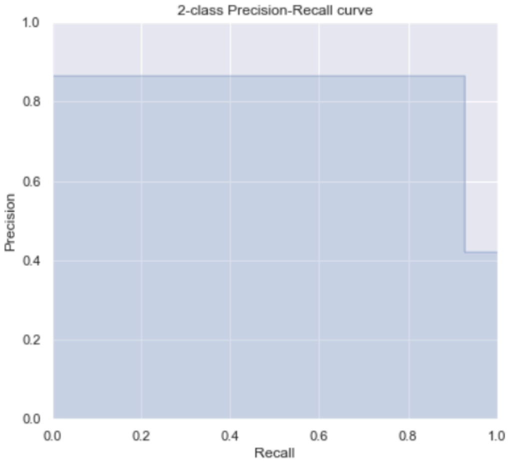


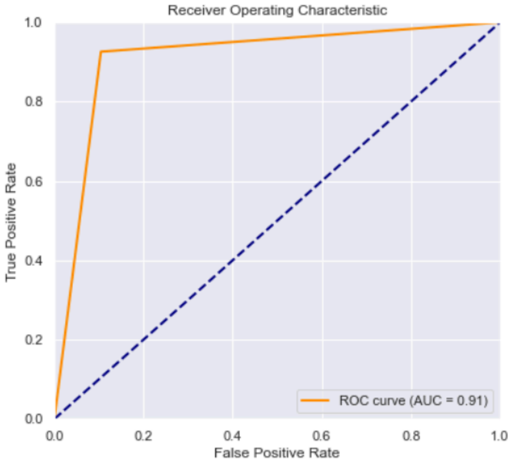
Boosting

Test Set

```
Accuracy : 0.9084 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, Worst: 0
Precision: 0.8662 [TP / (TP + FP)] Not to label a negative sample as positive.        Best: 1, Worst: 0
Recall   : 0.9260 [TP / (TP + FN)] Find all the positive samples.                     Best: 1, Worst: 0
ROC AUC  : 0.9108                                                                     Best: 1, Worst: < 0.5
-----------------------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of samples
```







Summary

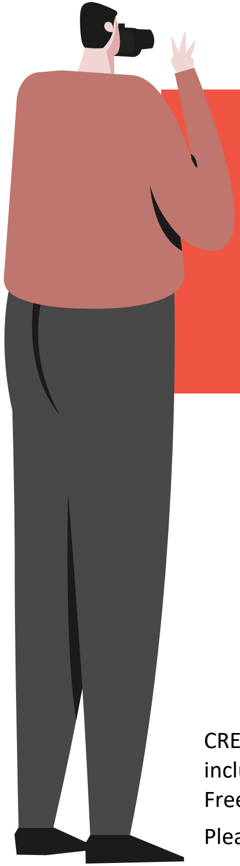
Training Models	Support Vector	Logistic Regression	Naïve Bayes	Boosting
Accuracy Scores (Training Sets)– 5 Text Columns	50.8%	49.3%	51.0%	50.**1**%
Accuracy Scores (Training Sets)– Identified 18 variables using forward feature selection	61.2%	74.6%	56.3%	**91.2%**

Evaluation

- Based on the data collected, we can use Machine Learning model — **Gradient Boosting** to help with the prediction of fake job listing. It provide the best accuracy at the rate of **91.2%.**
- AUC Score of **0.91** for both test and train test indicates a good classifier and overfitting is unlikely.
- Recruiters can use the predictive model to prevent such fraudulent job posters, improving the credibility of their recruitment domains.

Future Work...

- Try out other ML techniques and deep learning techniques to evaluate performance
- To deploy machine learning models to whole data frame. We could evaluate the accuracy, AUC score metrics if all the features were to be used as the predictor columns
- To present better models by using Grid Search to use the best estimators
- Resampling could have introduced bias. To test out variations of resampling with models
- To source for complementary and richer datasets from more recruitment sources to gather insightful data



THANKS!

Does anyone have any questions?

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

Please keep this slide for attribution.

RESOURCES

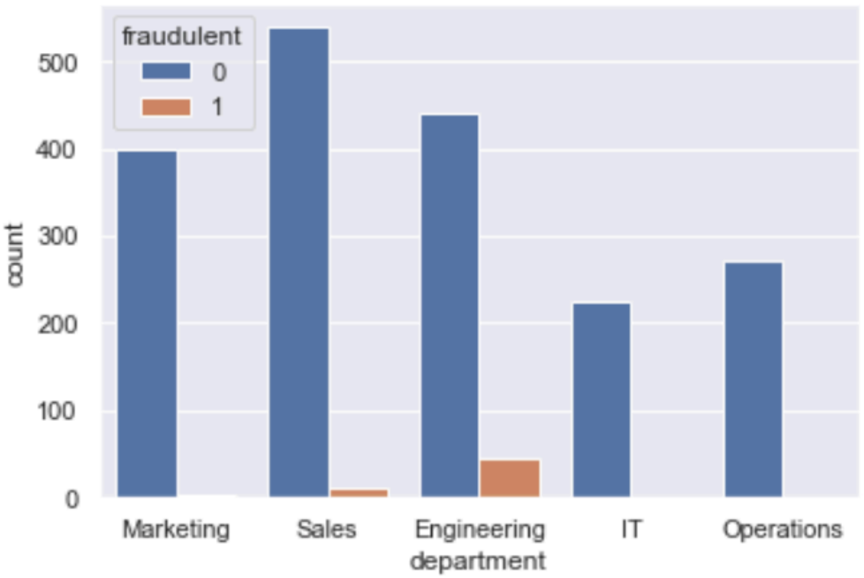
- https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd
- https://www.kdnuggets.com/2019/01/solve-90-nlp-problems-step-by-step-guide.html
- https://elitedatascience.com/imbalanced-classes
- https://www.cnbc.com/2020/10/06/job-scams-have-increased-during-the-covid-19-crisis-how-to-one.html

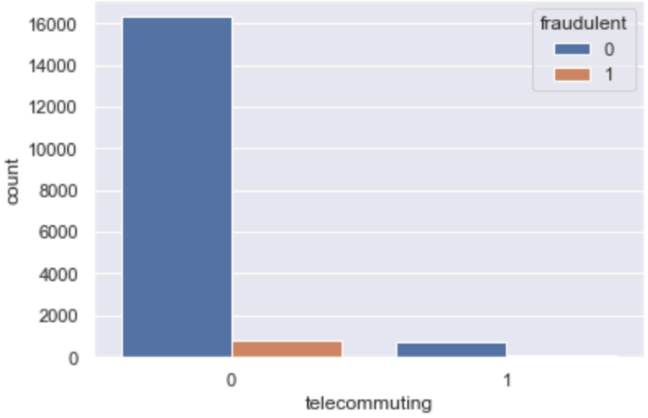
Appendices

Appendix

```
Job_postings['requirements']
```

```
0        Experience with content management systems a m...
1        What we expect from you:Your key responsibilit...
2        Implement pre-commissioning and commissioning ...
3        EDUCATION: Bachelor's or Master's in GIS, busi...
4        QUALIFICATIONS:RN license in the State of Texa...
                               ...
17875    To ace this role you:Will eat comprehensive St...
17876    - B.A. or B.S. in Accounting- Desire to have f...
17877    At least 12 years professional experience.Abil...
17878    1. Must be fluent in the latest versions of Co...
17879    We want to hear from you if:You have an in-dep...
Name: requirements, Length: 17880, dtype: object
```





Pre Processing of features

Encoding of categorical features

```
from sklearn.preprocessing import LabelEncoder
columns= ['telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
       'required_experience', 'required_education', 'industry', 'function']
label_encoder = LabelEncoder()
for i in columns:
    Job_postings[i] = label_encoder .fit_transform(Job_postings[i])
```

Basic text cleaning to each text column

```
def clean_text(company_profile):
    # reduce multiple spaces and newlines to only one
    company_profile = re.sub(r'(\s\s+|\n\n+)', r'\1', company_profile)
    # remove double quotes
    company_profile = re.sub(r'"', '', company_profile)
    # remove extra whitespace and special characters
    company_profile = re.sub(r'\s+$', '', company_profile)
    company_profile = re.sub(r'\^s+', '', company_profile)
    company_profile = re.sub(r'[^a-zA-Z\s]+', '', company_profile)
    #lowercase
    company_profile= company_profile.lower()
    # remove text between square brackets
    company_profile =re.sub('\[[^]]*\]', '',company_profile)


    # removes punctuation
    company_profile= re.sub(r'[^\w\s]','',company_profile)
    company_profile = re.sub(r'\n',' ',company_profile)
    company_profile= re.sub(r'[0-9]+','',company_profile)
    company_profile= re.sub(r'[0-9]+','',company_profile)
     # remove URLs
    company_profile = re.sub(r'https://t.co\S+\s*', '', company_profile)
    company_profile = re.sub(r'http://t.co\S+\s*', '', company_profile)

    return company_profile
```

Feature Extraction

```
Job_postings['word_count_r'] = Job_postings['requirements'].apply(lambda y: len(str(y).split(" ")))
Job_postings[['requirements','word_count_r']].head()
```

	requirements	word_count_r
5	experience with content management systems a m...	115
6	what we expect from youyour key responsibility...	187
7	implement precommissioning and commissioning p...	164
8	education bachelors or masters in gis business...	174
9	qualificationsrn license in the state of texas...	89

Further text cleaning

```python
def convert_text(description, remove_stop=True, lemma_words=False):

        # Remove stop words
    if remove_stop:
        description = description.split()
        description = [w for w in description if not w in stop]
        description = " ".join(description)



    #remove lemma_words
    if lemma_words:
        description = description.split()
        wl = nltk.stem.WordNetLemmatizer()
        lemma = ' '.join([wl.lemmatize(word) for word in description.split()])
    # tokenize
        description = nltk.word_tokenize(description)


    return description
```

```python
Job_postings.head()
```

nce	...	char_count_r	char_count_b	char_count_t	char_count_cp	word_density_d	word_density_cp	word_density_r	word_density_t	word_density_b	description1
5	...	715	0	15	710	6.064516	5.035461	6.217391	7.500000	0.000000	foodfast growingj ames...
7	...	1172	1022	35	909	5.333333	6.060000	6.267380	5.833333	4.542222	organise dfocuse dvibra...
0	...	1173	0	34	728	5.980000	5.352941	7.152439	8.500000	0.000000	clientlo catedho uston...
6	...	1228	669	28	509	6.414493	5.988235	7.057471	5.600000	6.968750	company esrienvi ronmen...
6	...	653	19	17	1327	7.142857	6.473171	7.337079	5.666667	6.333333	jobtitlei temizati onr...

Count Vectorization

```
X = Job_postings['title1']
y= fraudulent_upsampled['fraudulent'].sample(n=17880)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
# Learn a vocabulary dictionary of all tokens in the raw documents
counts.fit(X)

# Transform documents to document-term matrix.
X_train_count_1 = counts.transform(X_train)
X_test_count_1 = counts.transform(X_test)

print(X_train_count_1.shape)
print(X_test_count_1.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(14304, 26)
(3576, 26)
(14304,)
(3576,)
```

```
df_train_t=pd.DataFrame(X_train_count_1.todense(),columns=counts.get_feature_names())
df_test_t=pd.DataFrame(X_test_count_1.todense(),columns=counts.get_feature_names())
```

```
text_train_1 =pd.concat([df_train_d, df_train_t],axis=1)
text_test_1 =pd.concat([df_test_d, df_test_t],axis=1)
```

Confusion Matrices