Laporan Real/Fake Job Posting Detection

Martin Emmanuel Chang

Mikha Aldyn Yauw

Moses Anthony Kwik

Pixel Ariel Christopher

CALVIN INSTITUTE OF TECHNOLOGY

2024

DAFTAR ISI

1. Latar Belakang

Dalam era digital saat ini, di mana platform-platform digital untuk pencarian pekerjaan semakin populer, telah muncul masalah yang cukup meresahkan, yaitu adanya lowongan kerja palsu atau yang dikenal dengan istilah "fake job posting". Fenomena ini muncul seiring dengan pertumbuhan platform-platform digital tersebut. Lowongan kerja palsu dapat merugikan pencari pekerja dengan mengarahkan mereka ke penipuan atau pekerjaan yang tidak sesuai dengan deskripsi aslinya. Dalam kasus ini, pencari kerja mungkin akan menghabiskan waktu dan upaya untuk melamar pekerjaan yang sebenarnya tidak ada atau tidak nyata. Hal ini tidak hanya menimbulkan ketidaknyamanan, tetapi juga bisa berdampak finansial bagi pencari kerja.

Munculnya lowongan kerja palsu juga berdampak negatif pada reputasi platform pencarian kerja tersebut. Kepercayaan pencari kerja terhadap platform pencarian kerja dapat terganggu jika mereka terus-menerus menemui lowongan kerja palsu. Oleh karena itu, penting bagi platform-platform pencarian kerja untuk dapat mengidentifikasi dan menghapus lowongan kerja palsu dengan cepat dan efektif.

Untuk mengatasi masalah ini, kami mengusulkan solusi berupa sistem deteksi lowongan kerja palsu atau Fake Job Posting Detection. Sistem ini akan menggunakan teknik-teknik pemrosesan bahasa alami (Natural Language Processing/NLP) dan pembelajaran mesin (Machine Learning) untuk menganalisis teks dari deskripsi lowongan kerja dan menentukan apakah lowongan tersebut bersifat palsu atau tidak. Dengan demikian, sistem ini dapat memberikan perlindungan tambahan bagi pencari kerja dan membantu menjaga integritas platform-platform pencarian kerja.

1.1 Alasan Pemilihan Data

Kami memilih data dari website Kaggle karena dataset yang disediakan telah mencakup 17.014 data lowongan kerja yang nyata dan 866 data lowongan kerja palsu dari berbagai negara. Keputusan ini didasarkan pada keberagaman data yang tersedia, yang mencakup jumlah sampel yang signifikan dari kedua kategori, yaitu lowongan kerja nyata dan palsu. Hal ini penting untuk memastikan bahwa model yang akan kita bangun memiliki informasi yang cukup untuk dapat melakukan prediksi yang akurat dan dapat diandalkan.

Dengan menggunakan dataset ini, kami memiliki kesempatan untuk melakukan analisis yang mendalam terhadap pola-pola yang mungkin membedakan antara lowongan kerja nyata dan palsu. Data yang berasal dari berbagai negara juga memberikan gambaran yang lebih luas tentang fenomena lowongan kerja palsu di seluruh dunia. Ini memungkinkan kami untuk mengidentifikasi pola yang mungkin bersifat universal dan juga pola yang mungkin berbeda tergantung pada wilayah geografis.

Selain itu, dataset ini juga memberikan kesempatan bagi kami untuk mengembangkan model yang dapat memprediksi lowongan kerja palsu di masa depan. Dengan menggunakan teknik-teknik pembelajaran

mesin dan pemrosesan bahasa alami, kami dapat menghasilkan model yang dapat memeriksa dan menilai keaslian lowongan kerja baru yang ditambahkan ke platform pencarian kerja.

Dengan demikian, pemilihan dataset dari Kaggle ini memberikan landasan yang kokoh bagi kami untuk mengembangkan solusi deteksi lowongan kerja palsu yang efektif dan berkelanjutan, dengan tujuan akhir melindungi pencari kerja dari penipuan dan memperbaiki integritas platform pencarian kerja secara keseluruhan.

2. Pre-Processing

2.1 Data Manipulation

2.1.1 Feature Drop

Pre-processing Feature drop ini dilakukan dengan menghapus fitur yang korelasinya tidak terlalu signifikan terhadap fitur fraudulent melalui analisa heat map.
Hal ini dilakukan dengan cara berikut:

```
categorical=[]
numerical=[]

for col in df.columns:
    if df[col].dtypes !='object':
        numerical.append(col)
    else:
        categorical.append(col)
```

Pertama, kita bagi dulu data yang berbentuk categorical dan data yang berbentuk numerical.

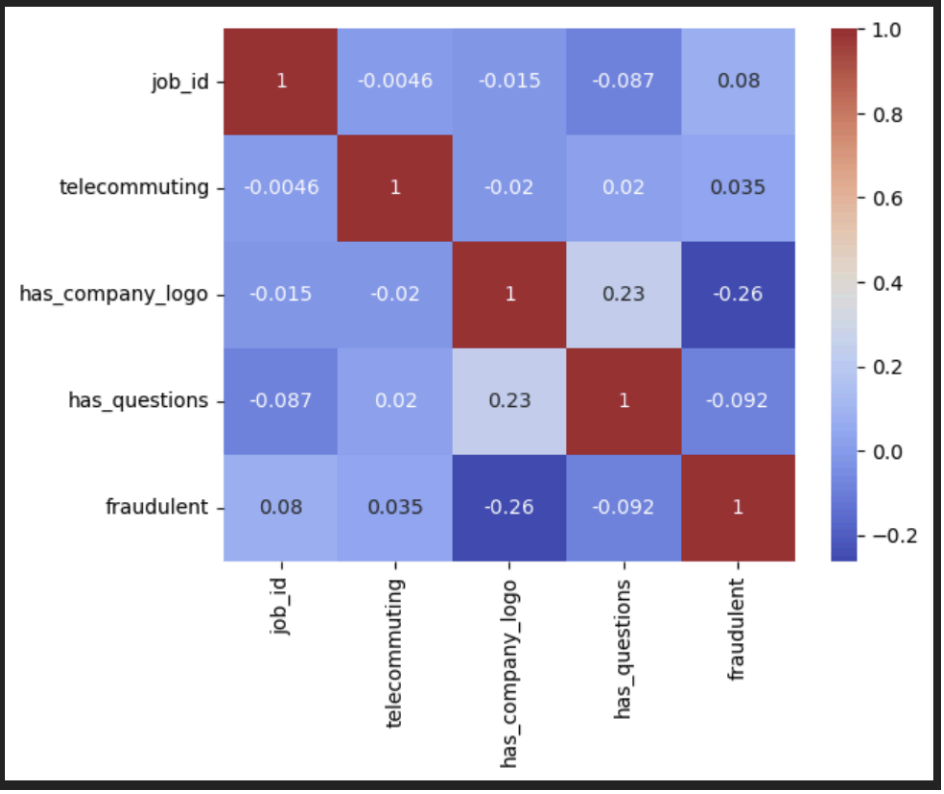
```
df[numerical].describe()
```

	job_id	telecommuting	has_company_logo	has_questions	fraudulent
count	17880.000000	17880.000000	17880.000000	17880.000000	17880.000000
mean	8940.500000	0.042897	0.795302	0.491723	0.048434
std	5161.655742	0.202631	0.403492	0.499945	0.214688
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	4470.750000	0.000000	1.000000	0.000000	0.000000
50%	8940.500000	0.000000	1.000000	0.000000	0.000000
75%	13410.250000	0.000000	1.000000	1.000000	0.000000
max	17880.000000	1.000000	1.000000	1.000000	1.000000

Dapat dilihat data yang berbentuk numerical.

```
sns.heatmap(df[numerical].corr(), cmap='coolwarm', annot=True)
```

```
<Axes: >
```



Lalu data-data numerical tersebut kami tampilkan dalam grafik heatmap. Kami menggunakan heatmap untuk mencari korelasi antar fitur dengan melihat nilai positive correlation dan negative correlationnya. Semakin nilai mendekati angka 1 maka korelasi positif semakin tinggi dimana artinya ketika suatu fitur meningkat maka fitur berkolerasi lainnya akan meningkat juga. Namun saat nilai suatu korelasi semakin mendekati angka -1 maka korelasi akan berbanding terbalik, artinya ketika suatu fitur meningkat maka fitur yang berkolerasi lainnya akan menurun. Oleh karena itu berdasarkan grafik heatmap diatas, kita dapat melihat bahwa fitur has_questions dan fitur telecommuting memiliki korelasi yang paling kecil dengan fitur fraudulent. Maka, kami memutuskan untuk menghapus fitur has_questions dan fitur telecommuting. Kami cukup mengambil fitur job_id dan has_company_logo saja.

```
numerical_to_drop=['telecommuting','has_questions']

df.drop(columns=numerical_to_drop, inplace=True)
```

Dengan kode berikut, maka kolom telecommuting dan has_questions telah berhasil dihapus dari tabel.

2.1.2 Missing Data Imputation

Pre-processing missing data imputation dilakukan dengan mengganti isi fitur yang berisi NaN menjadi isian string kosong atau “ “. Hal ini dilakukan agar seluruh data dapat diproses oleh komputer.

```
df.isnull().sum()
```

```
job_id                   0
title                    0
location               346
department           11547
salary_range         15012
company_profile       3308
description              1
requirements          2695
benefits              7210
telecommuting            0
has_company_logo         0
has_questions            0
employment_type       3471
required_experience   7050
required_education    8105
industry              4903
function              6455
fraudulent               0
dtype: int64
```

Jika dilihat pada gambar di atas, maka kita dapat melihat jumlah data null yang ada pada dataset sebelum di proses. Karena ada banyak data berjenis null, maka kita ingin melakukan proses missing data imputation dengan fungsi fillna.

```
#Missing data imputation

df.fillna(' ',inplace=True)
df.isnull().sum()
```

```
job_id                 0
title                  0
location               0
department             0
salary_range           0
company_profile        0
description            0
requirements           0
benefits               0
has_company_logo       0
employment_type        0
required_experience    0
required_education     0
industry               0
function               0
fraudulent             0
dtype: int64
```

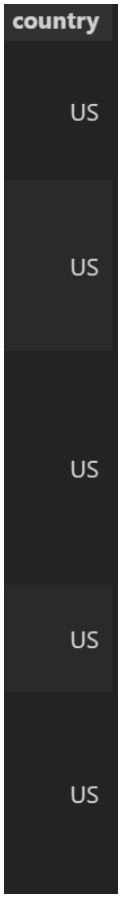
Setelah melakukan pemrosesan missing data imputation, maka dapat dilihat bahwa tidak ada data berjenis null pada dataset baru kita. Hal ini akan memudahkan pemrosesan data berikutnya.

2.1.3 Split Data

Split data berfungsi untuk mengambil data nama country dari fitur location. Lalu, data country tersebut dimasukkan ke kolom baru bernama country.

```
df['country'] = df['location'].str.split(',').str[0]
```

Kita melakukan splitting data menggunakan cara di atas. Kode ini akan memisahkan nama negara dari data location dan menyimpannya pada kolom baru bernama country. Hasil akan tampil seperti berikut:



2.1.4 Balancing & Sampling Data

Pre-processing balancing & sampling data bekerja dengan cara mengambil sampel data untuk menyeimbangkan jumlah data postingan pekerjaan yang asli dan palsu agar mudah diproses. Data asli akan dikurangi dan data palsu akan ditambahkan. Kami melakukan balancing dengan cara sampling agar data tidak bias ke data yang lebih banyak jumlahnya.

```
fake=df[df['fraudulent']==1]
fake.shape
```

```
(866, 17)
```

```
real=df[df['fraudulent']==0]
real.shape
```

```
(17014, 17)
```

Gambar di atas menunjukkan jumlah data fake dan real. Kami ingin mengubah jumlah kedua data ini menjadi seimbang, dengan cara sampling seperti berikut.

```
real=real.sample(3000,replace=True)
```

```
fake=fake.sample(3000,replace=True)
```

```
real.shape,fake.shape
```

```
((3000, 17), (3000, 17))
```

Setelah kami melakukan sampling, maka dapat terlihat bahwa data real dan fake sekarang sudah seimbang.

2.1.5 Feature Combinaton

Feature combination menggabungkan fitur yang datanya berupa string menjadi 1 fitur baru bernama combined_text. Kami menggunakan ini untuk memudahkan proses data cleaning, data normalization, dan feature extraction nantinya.

```
#Feature combination
```

```
df['combined_text'] = df[['title', 'location',
'salary_range','company_profile','description','requirements','benefits','
employment_type','required_experience','required_education','industry','fu
nction','department']].apply(lambda x: ' '.join(x), axis=1)



df.drop(columns=['title',

                 'location',

                 'salary_range',

                 'company_profile',

                 'description',

                 'requirements',

                 'benefits',

                 'employment_type',

                 'required_experience',

                 'required_education',

                 'industry',

                 'function',

                 'department'], inplace=True)

df.drop('country',axis=1,inplace=True)

df.head()
```

Kode di atas akan menghasilkan tabel, seperti berikut ini:

	job_id	has_company_logo	fraudulent	combined_text
0	6145	0	1	Physician Assistant PA Dermatology 1843 US, ...
1	17654	1	1	Senior JavaScript Developer US, IN, Bloomingto...
2	3609	1	1	Account Sales Managers $80-$130,000/yr US, NY,...
3	7666	0	1	Document Processing Assistant US, KS, Kansas ...
4	5692	0	1	Network Marketing US, DE, 7200-1380000 Are ...

Dapat dilihat bahwa kolom pada dataset sekarang tersisa 4 kolom saja, dimana data-data yang berupa string sudah tergabung pada satu kolom dengan nama combined_text.

2.2 Data Cleaning

Data cleaning bekerja dengan lowercasing, menghapus html tags, menghapus URLs, menghapus non-alphanumeric characters, tokenization, menghapus stopword, dan join token-token sebelumnya kembali menjadi string. Data cleaning membantu menyederhanakan text string yang ada. Kami menggunakan teknik ini karena kami hanya ingin mencari tahu kata-kata apa yang sering muncul pada poster lowongan kerja palsu maupun real. Oleh karena itu, kita tidak memerlukan tanda baca dan sebagainya.

```
def clean_text(text):

    text = text.lower()
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'[^a-zA-Z0-9]', ' ', text)
    tokens = nltk.word_tokenize(text)
    stop_words = set(nltk.corpus.stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    cleaned_text = ' '.join(tokens)

    return cleaned_text
```

Kami menggunakan fungsi clean_text di atas yang sudah berisi lowercasing text, penghapusan html tags, penghapusan URLs, penghapusan non-alphanumeric characters, tokenization, penghapusan stopword, dan join token-token sebelumnya kembali menjadi string.

```
df['combined_text']=df['combined_text'].apply(clean_text)
```

```
df.head()
```

	job_id	has_company_logo	fraudulent	combined_text
0	6145	0	1	physician assistant pa dermatology 1843 us ny ...
1	17654	1	1	senior javascript developer us bloomington rid...
2	3609	1	1	account sales managers 80 130 000 yr us ny uti...
3	7666	0	1	document processing assistant us ks kansas cit...
4	5692	0	1	network marketing us de 7200 1380000 looking m...

Dengan apply fungsi clean_text pada data combined_text, maka kita dapat menghasilkan suatu data string yang bersih dan sejenis.

2.3 Data Normalization

Data normalization bekerja dengan tokenisasi, lemmatisasi, dan penggabungan token kembali. Tokenisasi adalah pemisahan teks menjadi kata-kata individu. Lemmatisasi adalah proses mengubah kata ke bentuk dasarnya, misal walked menjadi walk. Terakhir, penggabungan token kembali menjadi satu string. Kami menggunakan teknik ini karena kami ingin mengganti semua kata menjadi bentuk bakunya. Sehingga, kata majemuk atau turunan yang memiliki akar kata yang sama, diklasifikasikan dalam satu klasifikasi yang sama. Contoh lemmatisasi:

```
Word: “walked”
Rule Application: Remove “-ed”
Result: “walk
```

Misal kita memiliki sebuah kalimat: The quick brown foxes are jumping over the lazy dogs.

Hasil lemmatisasi akan seperti berikut: the quick brown fox be jump over the lazy dog .

```python
nlp = spacy.load("en_core_web_sm")

def normalize_text(text):
    # Tokenize the text and apply lemmatization
    doc = nlp(text)
    normalized_words = [token.lemma_ for token in doc]
    normalized_text = ' '.join(normalized_words)
    return normalized_text
```

Kami melakukannya menggunakan fungsi normalize_text yang sudah dibuat seperti di atas.

```python
df['combined_text']=df['combined_text'].apply(normalize_text)
```

```python
print(df['combined_text'])
```

```
0       physician assistant pa dermatology 1843 us ny ...
1       senior javascript developer us bloomington rid...
2       account sale manager 80 130 000 yr we ny utica...
3       document processing assistant we ks kansas cit...
4       network market us de 7200 1380000 looking make...
                              ...
5995    sr php developer ph 07 cebu city zylun expand ...
5996    sale manager netigate oslo 03 oslo netigate le...
5997    section manager athens gr athens one uk lead r...
5998    graphic artist english service we dc washingto...
5999    construction manager us ct south windsor job o...
Name: combined_text, Length: 6000, dtype: object
```

Dengan apply fungsi normalize_text pada data combined_text, maka data yang sebelumnya telah dibersihkan dapat menghasilkan suatu data text yang baku.

2.4 Data Normalization

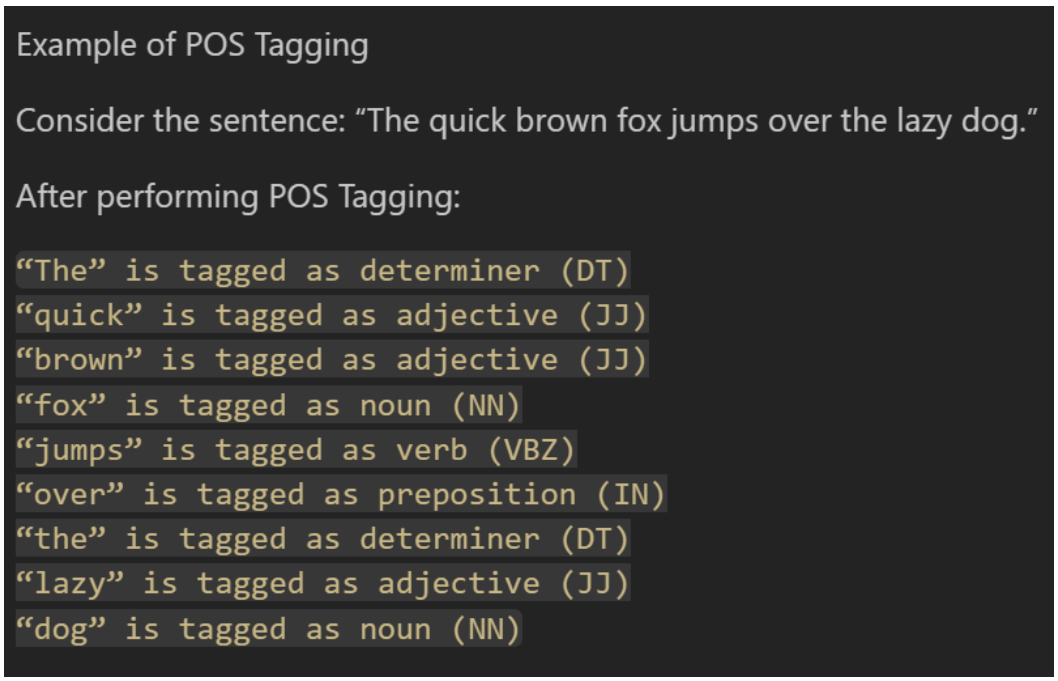
Feature extraction bekerja dengan cara pembuatan fitur POS (Part of Speech), penggabungan fitur POS, pembuatan matriks fitur, transformasi teks dan fitur POS, penggabungan matriks fitur, penyimpanan vectorizer.

- Pembuatan Fitur POS mengambil teks yang digabungkan, memecahnya menjadi token (kata-kata), dan kemudian memberi tag setiap token dengan part of speech-nya (misalnya, kata benda, kata kerja, dll.).
- Penggabungan Fitur POS mengambil daftar tupel tag POS dan menggabungkannya menjadi satu string.
- Pembuatan Matriks Fitur membuat objek CountVectorizer, yang akan mengubah teks menjadi matriks fitur. Setiap baris dalam matriks mewakili dokumen, dan setiap kolom mewakili fitur.
- Transformasi Teks dan Fitur POS mengubah teks dan fitur POS menjadi matriks fitur menggunakan CountVectorizer.
- Penggabungan Matriks Fitur menggabungkan dua matriks fitur menjadi satu.
- Penyimpanan Vectorizer menyimpan objek vectorizer ke file.

Secara keseluruhan, kode ini mengubah teks mentah dan fitur POS menjadi representasi numerik yang dapat dipahami oleh algoritma machine learning.

Kami menggunakan teknik ini karena kami ingin mengklasifikasi setiap kata sesuai part of speech-nya dan mengubahnya menjadi representasi numerik. Hal ini ditujukan untuk memudahkan komputer dalam membaca dan memproses data.

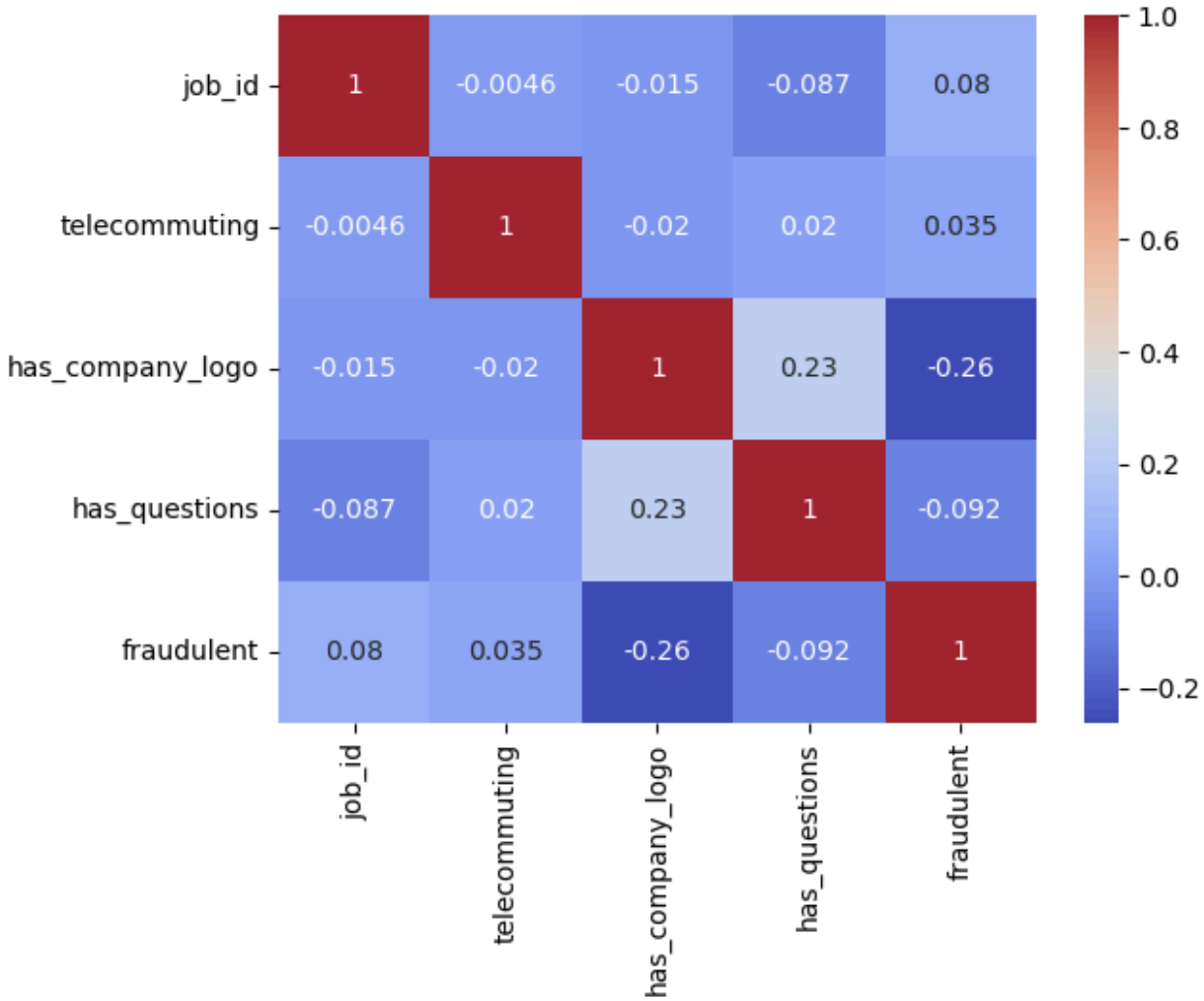
Contoh cara kerja POS Tagging:



3. Exploratory Data Analysis (EDA)

3.1 Heatmap

Heatmap adalah representasi visual data yang diplot sebagai warna dan angka dalam grid, untuk membantu identifikasi pola, hubungan, atau distribusi fitur dalam data dengan cepat dan intuitif.

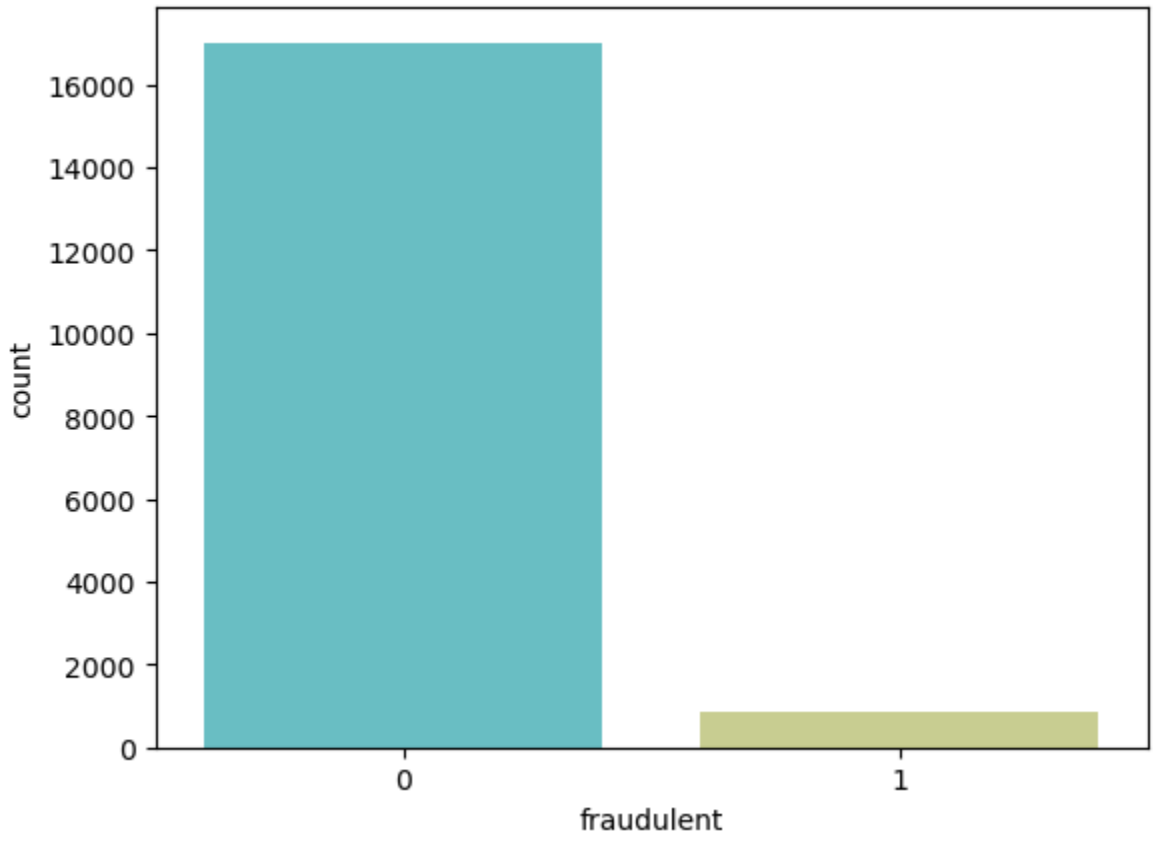


Heatmap di atas menunjukkan adanya korelasi antar fitur. Semakin mendekati nilai 1, korelasi positif antara dua fitur akan semakin tinggi, yang berarti jika satu fitur meningkat, kemungkinan besar fitur lainnya juga akan meningkat. Sebaliknya, semakin mendekati nilai -1, korelasi negatif antara dua fitur akan semakin kuat, yang mengindikasikan bahwa jika satu fitur meningkat, kemungkinan besar fitur yang lain akan menurun. Maka, kita dapat menentukan mana fitur-fitur yang memiliki korelasi yang sangat kecil satu dengan yang lainnya untuk tidak dipakai, sehingga tidak harus memakai semua fitur untuk efisiensi. Kami memutuskan untuk menghapus fitur has_questions dan fitur telecommuting, karena

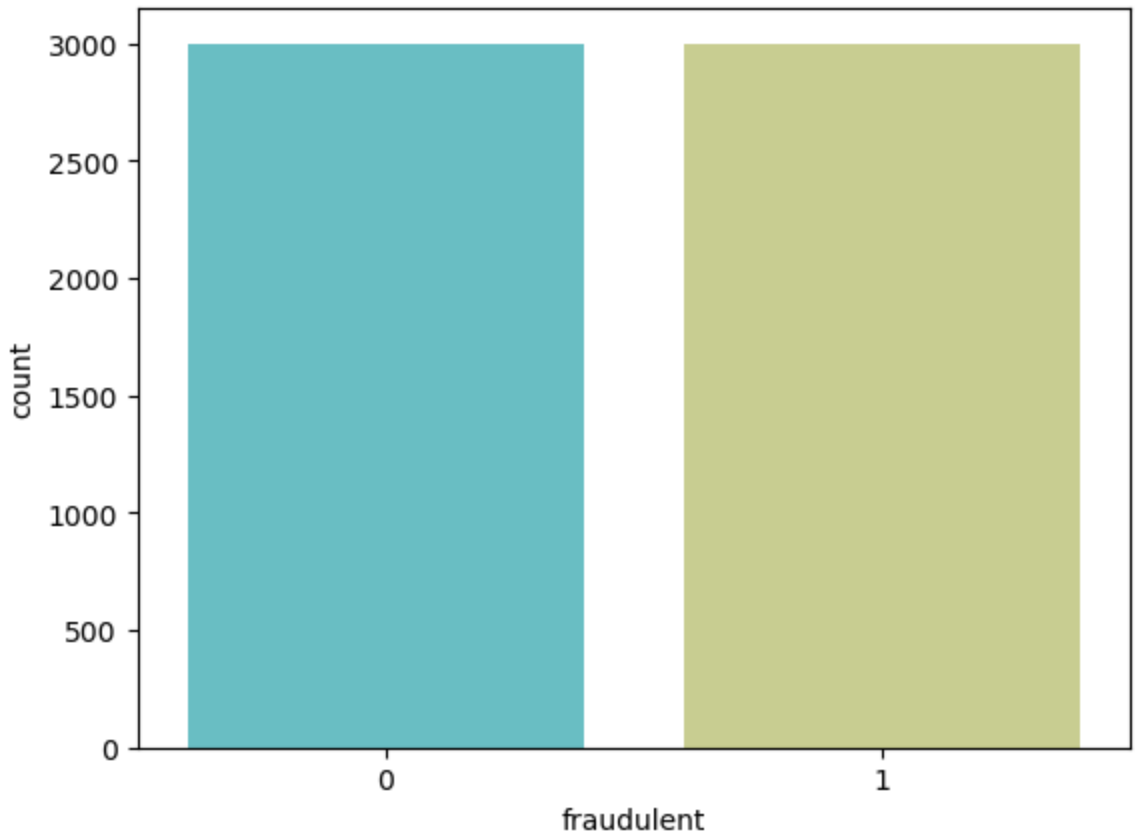
memiliki korelasi terkecil dengan fitur fraudulent. Fitur-fitur yang diambil untuk grafik heatmap ini adalah data-data numerical saja.

3.2 Bar Chart

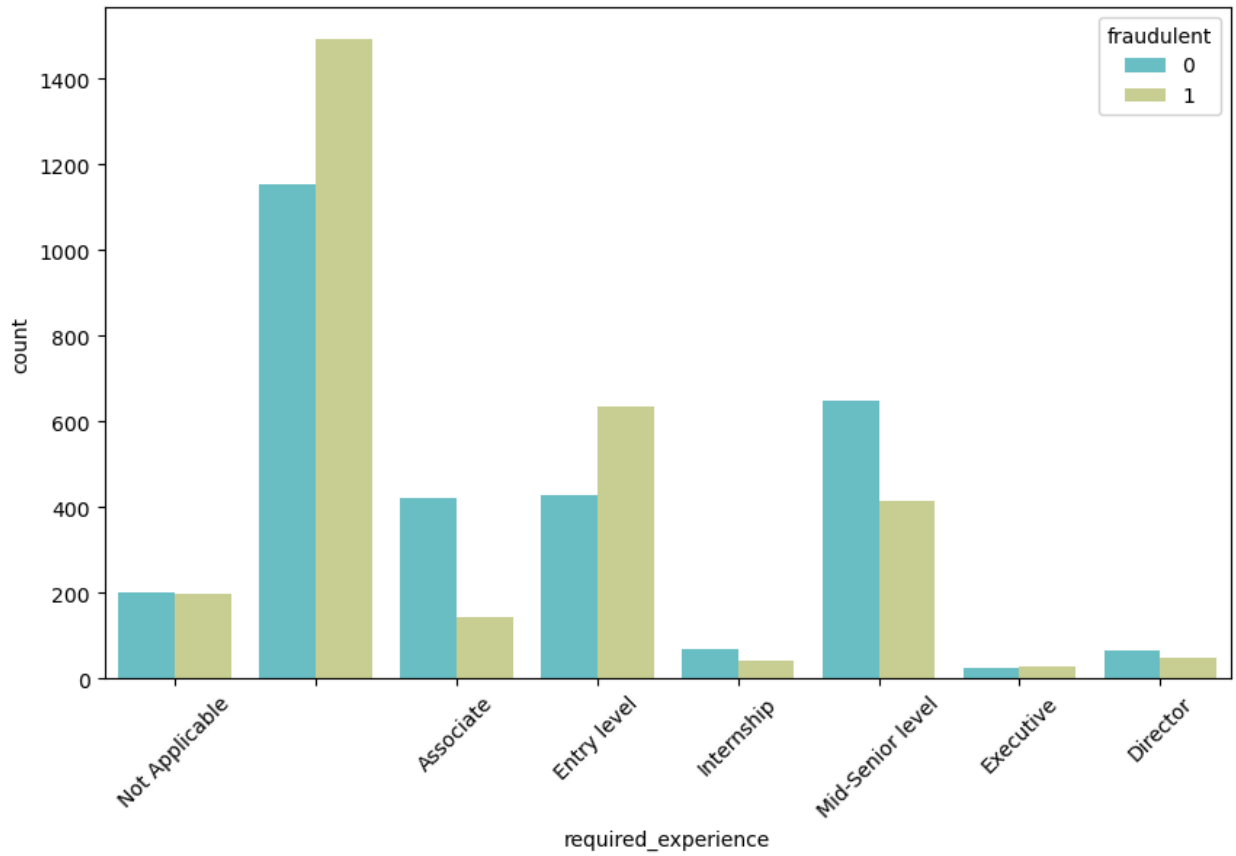
Diagram batang (bar chart) adalah jenis grafik yang menggunakan batang vertikal atau horizontal untuk memvisualisasikan data kuantitatif atau kualitatif. Setiap batang mewakili satu kategori atau variabel, dan tinggi (atau panjang) batang tersebut mencerminkan nilai atau frekuensi dari kategori tersebut. Diagram batang sering digunakan untuk membandingkan data antar kategori atau menunjukkan perubahan data dari waktu ke waktu.



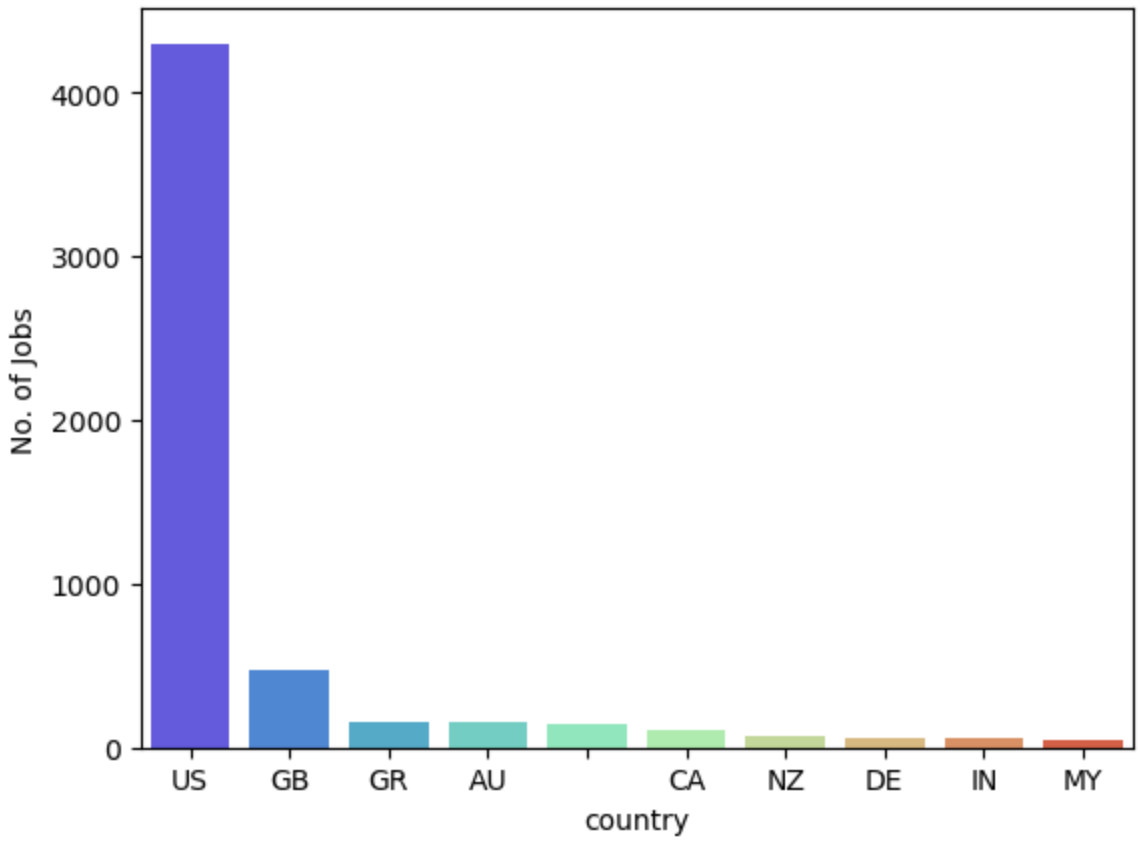
Gambar di atas memberi gambaran perbandingan jumlah data real dan fraudulent sebelum diproses dari dataset awal.



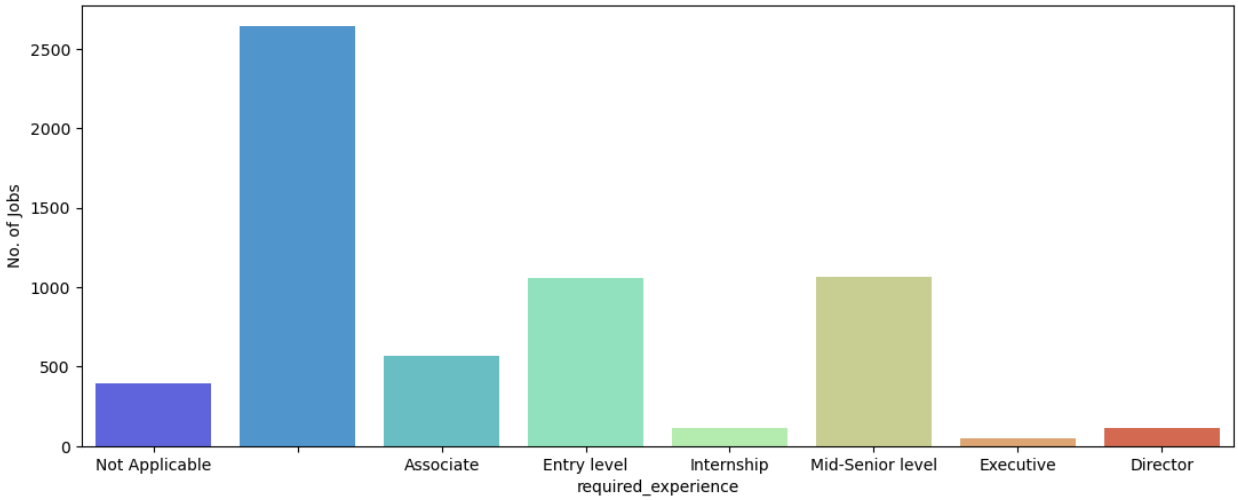
Gambar di atas ini menggambarkan perbandingan jumlah data real dan fraudulent setelah dibalance.



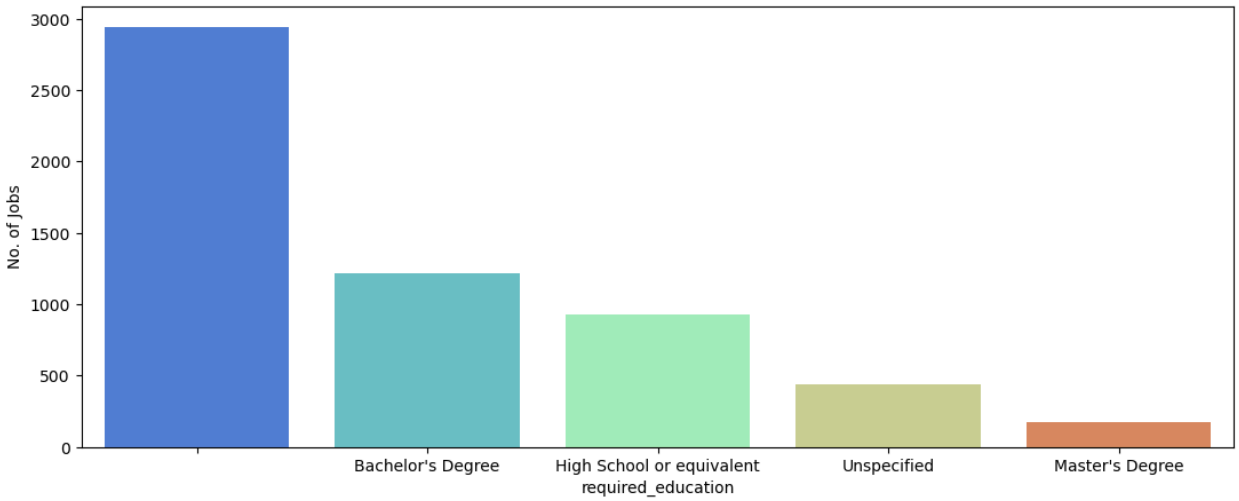
Gambar bar chart ini menunjukkan jumlah masing-masing posisi pada lowongan pekerjaan real dan fraudulent. Lowongan pekerjaan palsu paling banyak dan seringkali tidak menampilkan required experience yang dicari. Namun selain null, kita dapat lihat bahwa lowongan pekerjaan palsu seringkali memaparkan required experience di level entry. Dari sini kita dapat menyimpulkan bahwa pekerjaan yang paling mudah ditipu adalah pekerjaan entry level.



Gambar bar chart ini menunjukkan jumlah pekerjaan pada masing-masing negara. Dapat dilihat bahwa jumlah pekerjaan paling banyak ditawarkan di US.



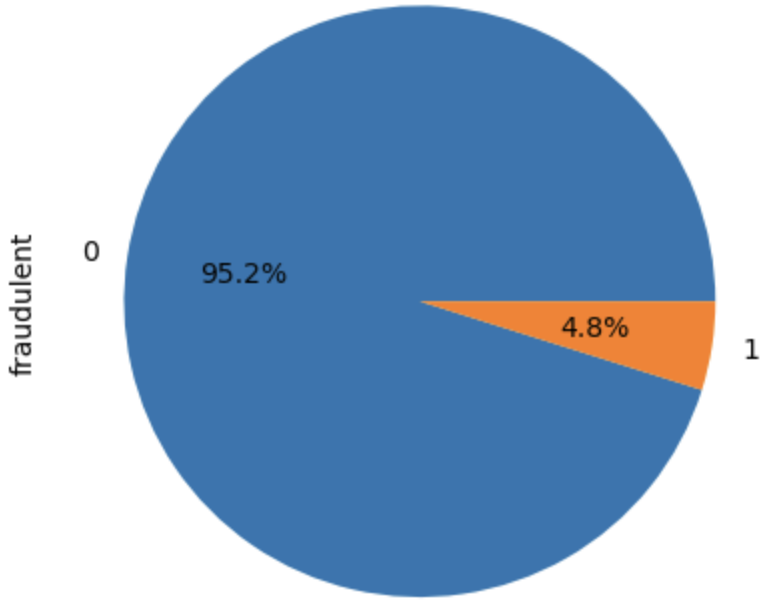
Gambar bar chart ini menunjukkan jumlah pekerjaan untuk setiap *required experience*. Dapat dilihat bahwa kebanyakan perusahaan tidak memiliki kriteria khusus terkait *required experience*.



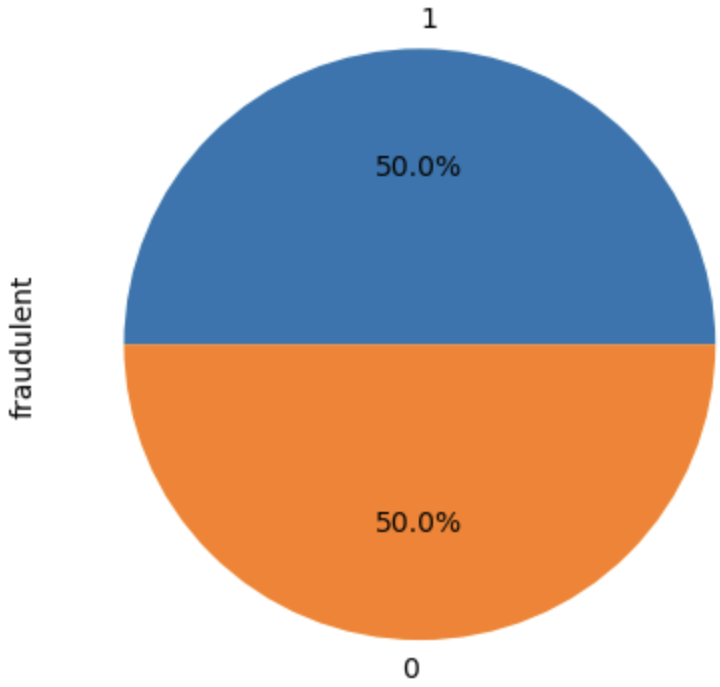
Gambar bar chart ini menunjukkan jumlah pekerjaan untuk setiap *required education*. Dapat dilihat bahwa kebanyakan perusahaan tidak memiliki kriteria khusus terkait *required education*. Namun, setelah itu rata-rata perusahaan meminta pelamarnya untuk memiliki *Bachelor's Degree*.

3.3 Pie Chart

Pie chart adalah jenis grafik yang menggunakan bentuk lingkaran untuk memvisualisasikan data kuantitatif atau proporsional. Lingkaran dibagi menjadi beberapa "iris" atau "potongan", di mana setiap potongan mewakili proporsi atau persentase dari total data. Potongan-potongan ini diletakkan berdampingan di sekitar lingkaran dan ukurannya berbanding lurus dengan besarnya nilai atau proporsi yang mereka wakili.



Gambar di atas memberi gambaran perbandingan jumlah data real dan fraudulent sebelum diproses dari dataset awal.

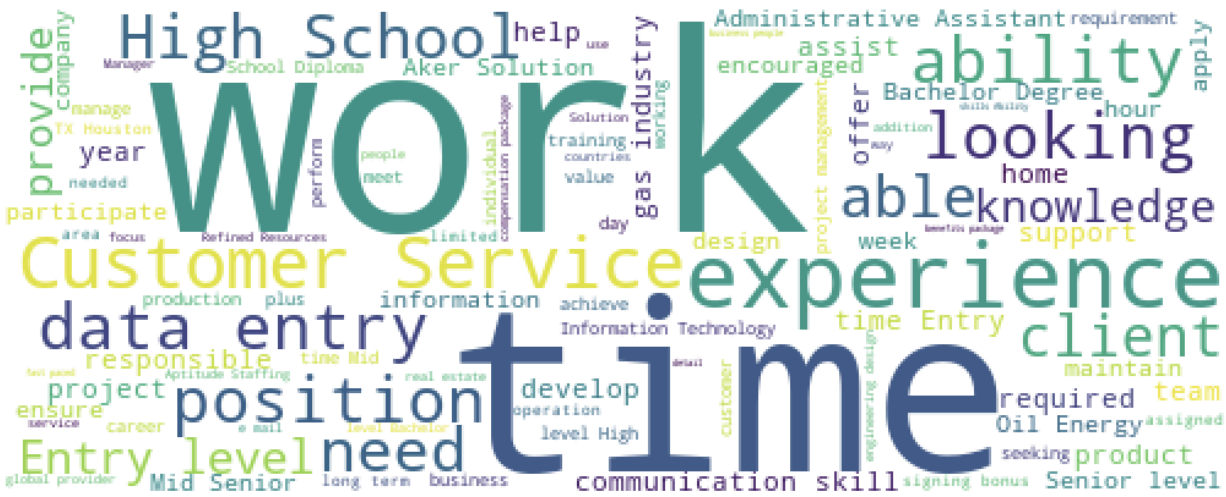


Gambar di atas ini menggambarkan perbandingan jumlah data real dan fraudulent setelah dibalance.

3.4 Word Cloud

Word cloud adalah representasi visual dari kumpulan kata-kata di mana kata-kata yang paling sering muncul diberi ukuran yang lebih besar dan lebih menonjol daripada kata-kata yang kurang sering muncul. Dalam word cloud, kata-kata disusun secara acak dan diatur dalam ukuran dan warna yang berbeda-beda untuk menciptakan visualisasi menarik yang memperlihatkan frekuensi relatif dari setiap kata dalam kumpulan teks. Word cloud sering digunakan untuk mengekstrak tema atau topik utama dari sebuah dokumen atau kumpulan teks, serta untuk memberikan gambaran umum tentang konten teks dengan cara yang mudah dimengerti secara visual.

Fake Word Cloud:



Gambar di atas melambangkan kata yang paling banyak digunakan pada iklan lowongan kerja fake.

Real Word Cloud:

Gambar di atas melambangkan kata yang paling banyak digunakan pada iklan lowongan kerja real.

3.5 Sparsity Pattern

Sparsity pattern adalah pola atau struktur kekosongan dalam sebuah matriks. Dalam konteks analisis data, sparsity pattern mengacu pada lokasi elemen-elemen nol dalam matriks data. Jika sebagian besar elemen matriks adalah nol, matriks tersebut dikatakan memiliki sparsity yang tinggi. Dalam pemodelan statistik, sparsity pattern dapat mengindikasikan bahwa hanya sebagian kecil dari variabel yang memiliki dampak signifikan terhadap hasil atau prediksi model. Oleh karena itu, identifikasi dan pemanfaatan sparsity pattern dapat membantu dalam seleksi fitur dan penyesuaian model yang lebih efisien.

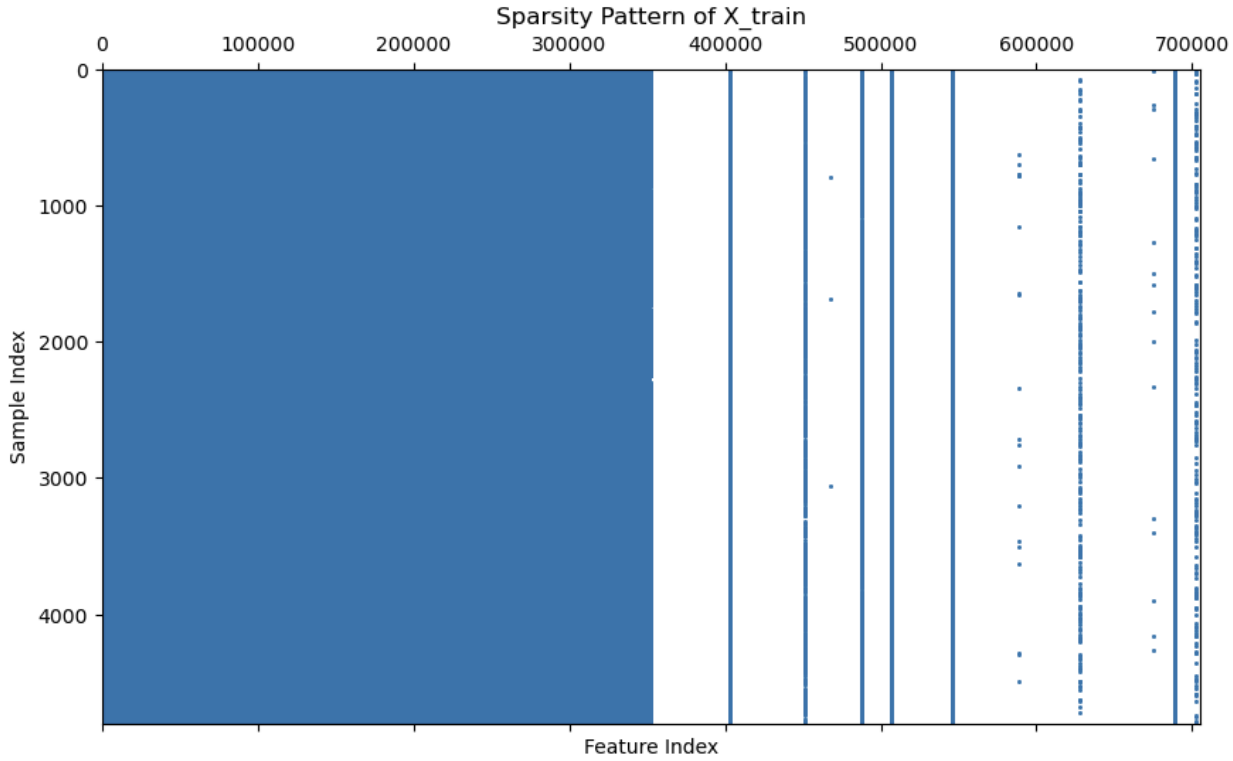
Sparsity pattern digunakan untuk memvisualisasikan data x train dan x test.

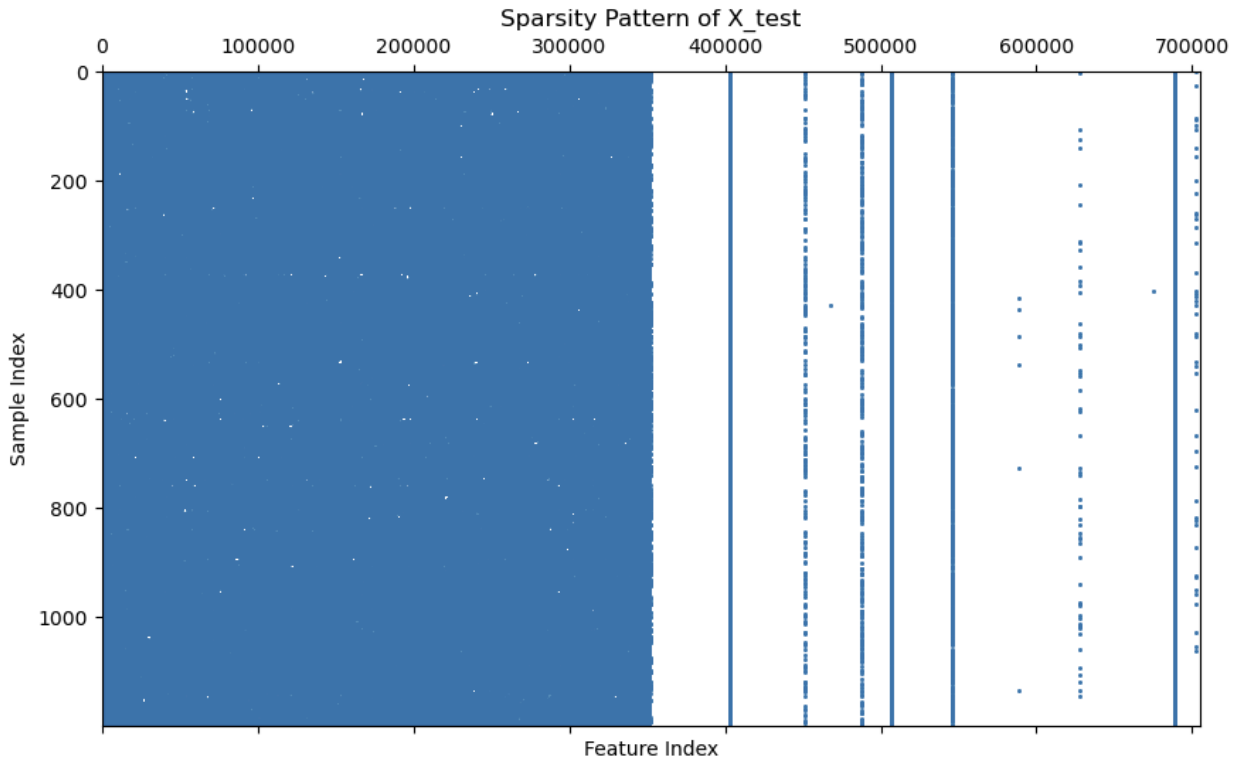
Cara kerja sparsity pattern seperti berikut:

Bayangkan kita memiliki sejumlah kotak, dan setiap kotak memiliki entri yang bisa berupa angka atau kata-kata. Jika ada kotak-kotak yang hanya berisi angka dan tidak ada kata-kata, kita bisa mengatakan bahwa kotak-kotak tersebut memiliki pola kekosongan yang terkait dengan angka. Sebaliknya, jika ada kotak-kotak yang hanya berisi kata-kata dan tidak ada angka, itu adalah pola kekosongan yang terkait dengan kata-kata.

Jadi, sparsity pattern dalam kasus ini adalah cara kita melihat di mana kotak-kotak di mana hanya angka yang ada dan di mana kotak-kotak di mana hanya kata-kata yang ada. Ini membantu kita memahami bagaimana data kita terorganisir dan apa yang ada di dalamnya. Dengan memahami pola kekosongan ini, kita bisa tahu di mana kita harus mencari angka dan di mana kita harus mencari kata-kata ketika kita ingin menggunakan atau menganalisis data kita.

Karena data kita sisa string dan integer, maka dengan visualisasi training dan testing data ini, kita dapat mudah mengetahui harus mencari string atau integer di bagian mana ketika kita ingin menggunakannya untuk analisa data.





4. Pelatihan dan Penyetelan Model Machine Learning

4.1 Kesesuaian pemilihan algoritma machine learning dengan masalah yang dihadapi.

Semua algoritma *Machine Learning* yang kami pilih adalah untuk klasifikasi karena proyek kami tentang *binary classification real/fake job posting.* Berikut cuplikan foto dari model yang kami gunakan dalam proyek ini.

```
models = [
    ('Logistic Regression', Pipeline([('scaler', StandardScaler(with_mean=False)), ('model', LogisticRegression(max_iter=500))])),
    ('Multinomial Naive Bayes', Pipeline([('model', MultinomialNB())])),
```

```
    ('RBF SVM', Pipeline([('scaler', StandardScaler(with_mean=False)), ('model',
SVC(kernel='rbf', gamma=2, C=1, random_state=42))])),
    ('Neural Network (MLP) Layer 1', Pipeline([('scaler', StandardScaler(with_mean=False)),
('model', MLPClassifier(hidden_layer_sizes=1, alpha=1, max_iter=2000, random_state=42))])),
    ('Extra Trees Classifier', Pipeline([('model', ExtraTreesClassifier(max_depth = 15,
n_estimators=100, random_state=42))])),
    ('Random Forest', Pipeline([('model', RandomForestClassifier(max_depth = 15,
random_state=42, n_estimators=100))]))
]
```

Kami menggunakan berbagai model agar dapat menentukan model mana yang paling sesuai dan mendapatkan kesimpulan sebagai berikut:

1. Logistic Regression:

- Kapan digunakan: Cocok untuk kasus klasifikasi biner (dua kelas).
- Kelebihan: Mudah diinterpretasi, cepat dilatih, cocok untuk data yang linier terpisah.
- Keterbatasan: Tidak cocok untuk masalah klasifikasi non-linier.

2. Multinomial Naive Bayes:

- Kapan digunakan: Cocok untuk klasifikasi teks atau data dengan fitur kategorikal yang dihitung dalam bentuk frekuensi, seperti pengklasifikasi email spam atau analisis sentimen pada teks.
- Kelebihan: Mudah diimplementasikan dan cepat dilatih, baik dalam menangani data dengan dimensi tinggi, serta cocok untuk data yang memiliki distribusi multinomial.
- Keterbatasan: Tidak cocok untuk masalah di mana fitur memiliki dependensi kompleks atau interaksi antar fitur.

3. Support Vector Machines (SVM):

- Kapan digunakan: Cocok untuk klasifikasi biner dengan margin linier atau non-linier yang besar.
- Kelebihan: Efektif dalam dimensi tinggi, dapat menangani masalah klasifikasi non-linier melalui kernel.
- Keterbatasan: Memerlukan tuning parameter, tidak cocok untuk set data besar.

4. Neural Network:

- Kapan digunakan: Cocok untuk klasifikasi dan regresi dalam masalah yang kompleks dan berdimensi tinggi, seperti gambar, teks, dan suara.
- Kelebihan: Mampu memodelkan pola yang sangat kompleks dalam data, bisa memberikan hasil yang baik dalam masalah yang tidak linear atau non-linear.

- Keterbatasan: Memerlukan jumlah data yang besar untuk pelatihan yang efektif, rentan terhadap overfitting, dan memerlukan komputasi yang besar, koneksi neural yang dibuat sulit untuk diinterpretasi.

5. Extra Trees:

- Kapan digunakan: Cocok untuk klasifikasi terlebih saat ingin membuat model yang lebih general dan memerlukan waktu yang cepat untuk dilatih.
- Kelebihan: Lebih efisien daripada Random Forest karena menggunakan sampel acak dalam proses pemilihan fitur dan pembentukan pohon, mampu mengatasi overfitting, dan cenderung lebih cepat dalam pelatihan.
- Keterbatasan: Tidak mudah diinterpretasi seperti RF atau Decision Tree.

6. Random Forest:

- Kapan digunakan: Cocok untuk klasifikasi dan dapat menangani banyak fitur.
- Kelebihan: Mengurangi overfitting melalui keputusan beberapa pohon.
- Keterbatasan: Tidak mudah diinterpretasi seperti Decision Tree.

4.2 Membagi data menjadi data untuk pelatihan dan data untuk diuji

Kami telah membagi data untuk pelatihan (*training dataset*) sebesar 0.8 dan data untuk di uji (*test dataset*) sebesar 0.2 agar menghasilkan model lebih tergeneralisasi dan tidak *over-fitting* pada data yang ada serta mampu memprediksi *unseen data*.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4.3 Menggunakan 3 algoritma berbeda

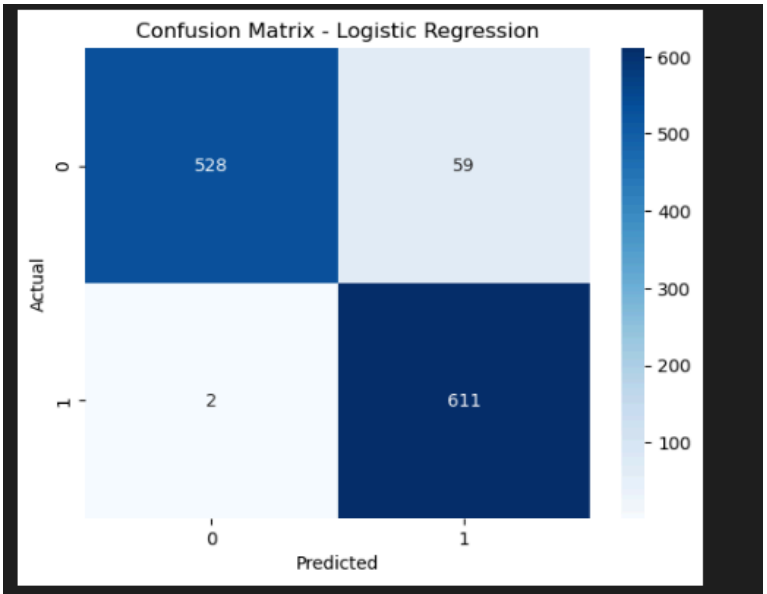
1. Logistic Regression
2. SVM
3. Random Forest

4.4 Menggunakan 1 algoritma yang belum pernah diajarkan

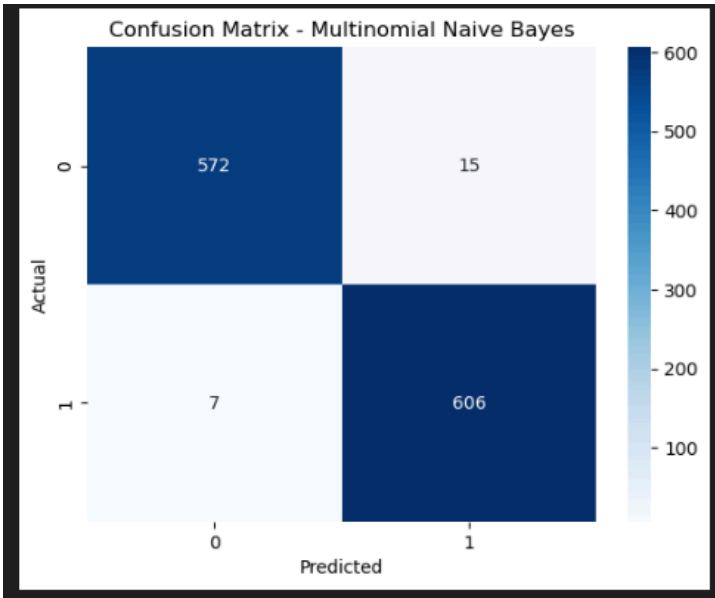
1. Extra Trees Classifier dan Neural Network (Multi Layer Perceptron)

5. Performance Measuring

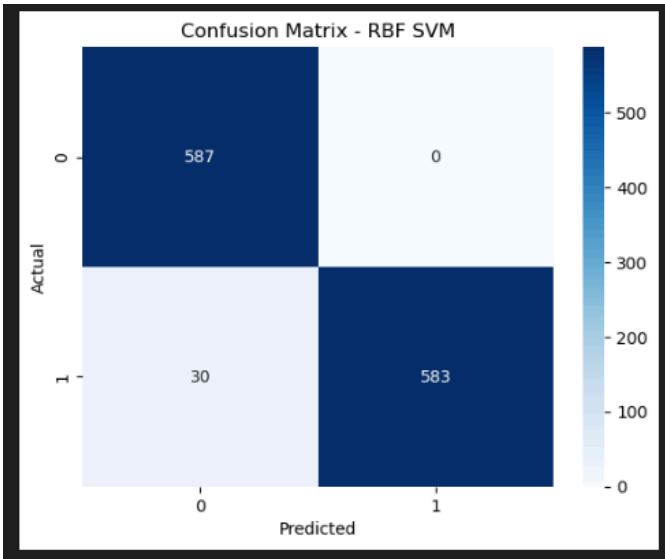
5.1 Cross-Validation, K-Fold Validation, Stratified K-Fold Validation



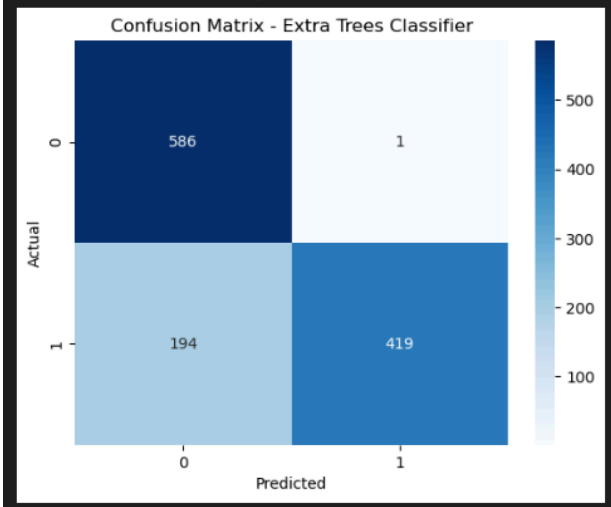

```
Model: Logistic Regression
Test set accuracy: 0.9491666666666667
Test set precision: 0.9540833568009012
Test set recall: 0.9481131420027735
Test set F1 score: 0.9489223095768576
Cross-Validation Scores: [0.955      0.9575     0.94416667 0.94916667 0.9475
Mean CV Score: 0.9506666666666668
K-Fold 1 Accuracy: 0.9491666666666667
K-Fold 2 Accuracy: 0.9516666666666667
K-Fold 3 Accuracy: 0.9433333333333334
K-Fold 4 Accuracy: 0.9433333333333334
K-Fold 5 Accuracy: 0.9525
Stratified K-Fold 1 Accuracy: 0.9558333333333333
Stratified K-Fold 2 Accuracy: 0.9516666666666667
Stratified K-Fold 3 Accuracy: 0.9491666666666667
Stratified K-Fold 4 Accuracy: 0.945
Stratified K-Fold 5 Accuracy: 0.9541666666666667
```



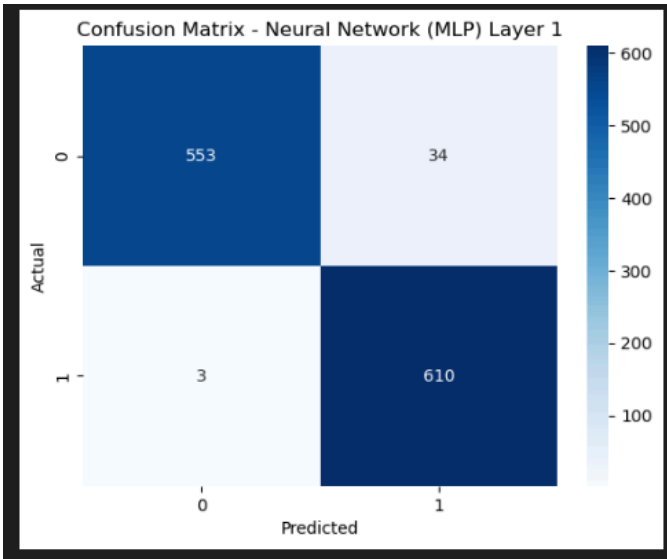

```
Model: Multinomial Naive Bayes
Test set accuracy: 0.9816666666666667
Test set precision: 0.9818778003053741
Test set recall: 0.9815135438580889
Test set F1 score: 0.9816519372496254
Cross-Validation Scores: [0.98333333 0.985      0.98416667 0.97833333 0.9775
Mean CV Score: 0.9816666666666667
K-Fold 1 Accuracy: 0.9816666666666667
K-Fold 2 Accuracy: 0.9825
K-Fold 3 Accuracy: 0.9775
K-Fold 4 Accuracy: 0.9816666666666667
K-Fold 5 Accuracy: 0.9833333333333333
Stratified K-Fold 1 Accuracy: 0.9808333333333333
Stratified K-Fold 2 Accuracy: 0.9808333333333333
Stratified K-Fold 3 Accuracy: 0.9816666666666667
Stratified K-Fold 4 Accuracy: 0.9766666666666667
Stratified K-Fold 5 Accuracy: 0.9833333333333333
```



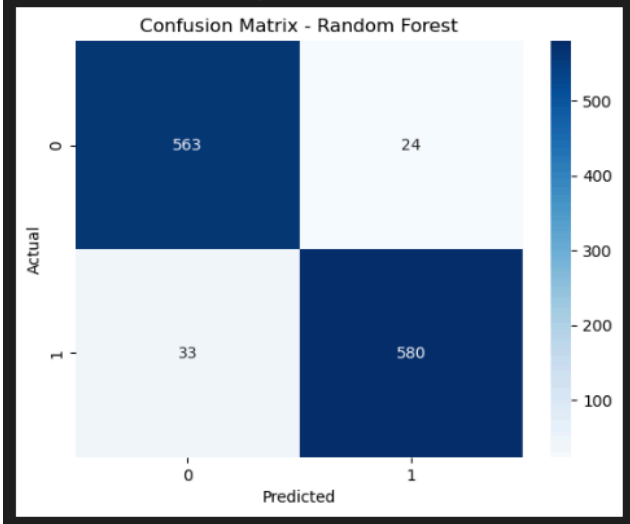

```
Model: RBF SVM
Test set accuracy: 0.975
Test set precision: 0.9756888168557536
Test set recall: 0.9755301794453508
Test set F1 score: 0.9749997222191358
Cross-Validation Scores: [0.96416667 0.9725     0.9775     0.96333333 0.97
Mean CV Score: 0.9695
K-Fold 1 Accuracy: 0.975
K-Fold 2 Accuracy: 0.97
K-Fold 3 Accuracy: 0.9666666666666667
K-Fold 4 Accuracy: 0.9625
K-Fold 5 Accuracy: 0.9758333333333333
Stratified K-Fold 1 Accuracy: 0.9758333333333333
Stratified K-Fold 2 Accuracy: 0.9683333333333334
Stratified K-Fold 3 Accuracy: 0.9683333333333334
Stratified K-Fold 4 Accuracy: 0.975
Stratified K-Fold 5 Accuracy: 0.9683333333333334
```




```
Model: Extra Trees Classifier
Test set accuracy: 0.8375
Test set precision: 0.8744505494505495
Test set recall: 0.8409100383235464
Test set F1 score: 0.8342906471233493
Cross-Validation Scores: [0.84666667 0.84166667 0.83833333 0.84583333 0.84916667]
Mean CV Score: 0.8443333333333334
K-Fold 1 Accuracy: 0.8375
K-Fold 2 Accuracy: 0.8566666666666667
K-Fold 3 Accuracy: 0.8458333333333333
K-Fold 4 Accuracy: 0.8358333333333333
K-Fold 5 Accuracy: 0.8725
Stratified K-Fold 1 Accuracy: 0.8541666666666666
Stratified K-Fold 2 Accuracy: 0.8666666666666667
Stratified K-Fold 3 Accuracy: 0.8641666666666666
Stratified K-Fold 4 Accuracy: 0.8541666666666666
Stratified K-Fold 5 Accuracy: 0.8558333333333333
```




```
Model: Neural Network (MLP) Layer 1
Test set accuracy: 0.9691666666666666
Test set precision: 0.970904642745431
Test set recall: 0.9685922002273288
Test set F1 score: 0.9690969416412447
Cross-Validation Scores: [0.97416667 0.97583333 0.96416667 0.96833333 0.97083333]
Mean CV Score: 0.9706666666666667
K-Fold 1 Accuracy: 0.9683333333333334
K-Fold 2 Accuracy: 0.975
K-Fold 3 Accuracy: 0.9641666666666666
K-Fold 4 Accuracy: 0.97
K-Fold 5 Accuracy: 0.9808333333333333
Stratified K-Fold 1 Accuracy: 0.98
Stratified K-Fold 2 Accuracy: 0.9675
Stratified K-Fold 3 Accuracy: 0.9708333333333333
Stratified K-Fold 4 Accuracy: 0.9641666666666666
Stratified K-Fold 5 Accuracy: 0.9716666666666667
```




```
Model: Random Forest
Test set accuracy: 0.9525
Test set precision: 0.9524478865727366
Test set recall: 0.9526402672365638
Test set F1 score: 0.9524904651002875
Cross-Validation Scores: [0.9575     0.9525     0.95416667 0.94416667 0.94833333]
Mean CV Score: 0.9513333333333334
K-Fold 1 Accuracy: 0.9566666666666667
K-Fold 2 Accuracy: 0.955
K-Fold 3 Accuracy: 0.94
K-Fold 4 Accuracy: 0.9533333333333334
K-Fold 5 Accuracy: 0.9483333333333334
Stratified K-Fold 1 Accuracy: 0.9558333333333333
Stratified K-Fold 2 Accuracy: 0.9475
Stratified K-Fold 3 Accuracy: 0.9366666666666666
Stratified K-Fold 4 Accuracy: 0.9525
Stratified K-Fold 5 Accuracy: 0.9525
```

5.2 Accuracy, Precision, Recall, F1-Score Test Set

	Model	Test set accuracy	Test set precision	Test set recall	Test set F1 score	Confusion matrix
0	Logistic Regression	0.949167	0.954083	0.948113	0.948922	[[528, 59], [2, 611]]
1	Multinomial Naive Bayes	0.981667	0.981878	0.981514	0.981652	[[572, 15], [7, 606]]
2	RBF SVM	0.975000	0.975689	0.975530	0.975000	[[587, 0], [30, 583]]
3	Neural Network (MLP) Layer 1	0.969167	0.970905	0.968592	0.969097	[[553, 34], [3, 610]]
4	Extra Trees Classifier	0.837500	0.874451	0.840910	0.834291	[[586, 1], [194, 419]]
5	Random Forest	0.952500	0.952448	0.952640	0.952490	[[563, 24], [33, 580]]

K-fold: K-fold cross-validation adalah metode untuk mengevaluasi kinerja model machine learning dengan membagi data menjadi kumpulan "lipatan" atau "fold" sebanyak K bagian yang sama besar. Kita menggunakan tekniki ini untuk menghindari overfitting dan underfitting karena menggunakan setiap bagian data untuk validasi dan pelatihan.

Stratified K-fold: Stratified K-fold cross-validation adalah varian dari K-fold cross-validation yang memastikan distribusi kelas yang seimbang di setiap lipatan. Ini sangat penting ketika data memiliki ketidakseimbangan kelas, di mana jumlah sampel dalam setiap kelas tidak merata. Kita menggunakan ini untuk memastikan bahwa setiap lipatan mencerminkan proporsi kelas yang sama dengan dataset asli.

Cross Val Score: Cross-validation score adalah metrik evaluasi yang digunakan untuk mengukur kinerja model pada data yang tidak terlihat. Ini merupakan hasil dari proses cross-validation, biasanya berupa skor rata-rata dari skor validasi setiap lipatan. Kita menggunakan model ini untuk mendapatkan perkiraan yang lebih baik tentang kinerja model di luar data pelatihan.

Accuracy: Akurasi adalah rasio prediksi yang benar secara keseluruhan terhadap jumlah total prediksi. Ini adalah salah satu metrik evaluasi yang paling umum digunakan dan mudah dipahami, terutama untuk masalah klasifikasi biner.

Precision: Presisi adalah ukuran dari seberapa banyak item yang diprediksi positif oleh model yang benar-benar positif.
Ini memberi tahu kita seberapa banyak prediksi positif yang benar dibandingkan dengan semua prediksi positif yang dilakukan oleh model.

F1-Score: F1-Score adalah rata-rata harmonik dari recall dan precision. Ini memberi kita pengukuran tunggal tentang seimbang antara recall dan precision.

Recall: Recall adalah ukuran dari seberapa banyak item positif yang ditemukan oleh model dari semua item positif yang sebenarnya. Ini memberi tahu kita seberapa banyak item positif yang berhasil diidentifikasi oleh model.

DAFTAR PUSTAKA

https://www.datacamp.com/blog/classification-machine-learning

https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501

https://www.kaggle.com/code/seifwael123/real-fake-jobs-eda-modelling-99/notebook