In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import classification_report
from collections import defaultdict

# Load the dataset
file_path = '/Users/akshaythakare/Downloads/fake_job_postings.csv'
data = pd.read_csv(file_path)
```

In [3]:

```
data.isnull().sum()
```

Out[3]:

```
job_id                     0
title                      0
location                 346
department             11547
salary_range           15012
company_profile         3308
description                1
requirements            2696
benefits                7212
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
```

In [5]:

```
# Fill missing values in text columns with an empty string
text_columns = ['title', 'location', 'department', 'company_profile', 'descri
for col in text_columns:
    data[col] = data[col].fillna('')
```

In [7]:
```python
data.isnull().sum()
```

Out[7]:
```
job_id                     0
title                      0
location                   0
department                 0
salary_range           15012
company_profile            0
description                0
requirements               0
benefits                   0
telecommuting              0
has_company_logo           0
has_questions              0
employment_type         3471
required_experience     7050
required_education      8105
industry                   0
function                6455
fraudulent                 0
dtype: int64
```

In [9]:
```python
# Create a binary indicator for missing salary values
data['salary_missing'] = data['salary_range'].isnull().astype(int)

# Define a function to process the salary range into a numeric format
def process_salary(salary):
    if pd.isnull(salary):
        return np.nan
    salary_range = salary.split('-')
    try:
        return (float(salary_range[0]) + float(salary_range[1])) / 2
    except:
        return np.nan

# Apply the function to convert salary range to numeric
data['salary'] = data['salary_range'].apply(process_salary)

# Impute salary based on experience level, then education level, and finally
data['salary'] = data.groupby('required_experience')['salary'].transform(lamb
data['salary'] = data.groupby('required_education')['salary'].transform(lambd
data['salary'] = data['salary'].fillna(data['salary'].median())

# Drop the original 'salary_range' column as it's now converted
data.drop(columns=['salary_range'], inplace=True)
```

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
job_id                    0
title                     0
location                  0
department                0
company_profile           0
description               0
requirements              0
benefits                  0
telecommuting             0
has_company_logo          0
has_questions             0
employment_type        3471
required_experience    7050
required_education     8105
industry                  0
function               6455
fraudulent                0
salary_missing            0
salary                    0
dtype: int64
```

In [13]:
```
data['employment_type'].fillna(data['employment_type'].mode()[0], inplace=Tru
```

```
/var/folders/2m/6n8651_j1vz7qclc1y39fx8c0000gn/T/ipykernel_35309/259688842
6.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behav
es as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) in
stead, to perform the operation inplace on the original object.


  data['employment_type'].fillna(data['employment_type'].mode()[0], inplace
=True)
```

In [15]:

```
data.isnull().sum()
```

Out[15]:

```
job_id                   0
title                    0
location                 0
department               0
company_profile          0
description              0
requirements             0
benefits                 0
telecommuting            0
has_company_logo         0
has_questions            0
employment_type          0
required_experience   7050
required_education    8105
industry                 0
function              6455
fraudulent               0
salary_missing           0
salary                   0
dtype: int64
```

In [17]:

```
# Convert required experience to numeric value
def process_experience(experience):
    if pd.isnull(experience):
        return np.nan
    exp_range = experience.split('-')
    try:
        return (float(exp_range[0]) + float(exp_range[1])) / 2
    except:
        return np.nan

data['experience'] = data['required_experience'].apply(process_experience)

# Impute missing experience values based on the median within each education
data['experience'] = data.groupby('required_education')['experience'].transfo
data['experience'] = data['experience'].fillna(data['experience'].median())
```

In [19]:

```
data['experience'] = data['experience'].fillna(data['experience'].median())
```

In [21]:

```python
# Fill missing education values with the mode
data['required_education'].fillna(data['required_education'].mode()[0], inpla
```

```
/var/folders/2m/6n8651_j1vz7qclc1y39fx8c0000gn/T/ipykernel_35309/126041744
4.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behav
es as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) in
stead, to perform the operation inplace on the original object.


  data['required_education'].fillna(data['required_education'].mode()[0], i
nplace=True)
```

In [23]:

```python
data['function'].fillna(data['function'].mode()[0], inplace=True)
```

```
/var/folders/2m/6n8651_j1vz7qclc1y39fx8c0000gn/T/ipykernel_35309/799071214.
py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame o
r Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behav
es as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) in
stead, to perform the operation inplace on the original object.


  data['function'].fillna(data['function'].mode()[0], inplace=True)
```

In [25]:

```python
# Impute missing 'experience' values based on education level, then employmen
data['experience'] = data.groupby('required_education')['experience'].transfo
data['experience'] = data.groupby('employment_type')['experience'].transform(

# If there are still missing values, fill them with the overall median of 'ex
data['experience'].fillna(data['experience'].median(), inplace=True)
```

```
/var/folders/2m/6n8651_j1vz7qclc1y39fx8c0000gn/T/ipykernel_35309/324341259
4.py:6: FutureWarning: A value is trying to be set on a copy of a DataFrame
or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behav
es as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) in
stead, to perform the operation inplace on the original object.


  data['experience'].fillna(data['experience'].median(), inplace=True)
```

In [27]:
```
data.isnull().sum()
```

Out[27]:
```
job_id                    0
title                     0
location                  0
department                0
company_profile           0
description               0
requirements              0
benefits                  0
telecommuting             0
has_company_logo          0
has_questions             0
employment_type           0
required_experience    7050
required_education        0
industry                  0
function                  0
fraudulent                0
salary_missing            0
salary                    0
experience            17880
dtype: int64
```

In [29]:
```
data.drop(columns=['experience','required_experience'], inplace=True)
```

In [31]:
```
data.isnull().sum()
```

Out[31]:
```
job_id                0
title                 0
location              0
department            0
company_profile       0
description           0
requirements          0
benefits              0
telecommuting         0
has_company_logo      0
has_questions         0
employment_type       0
required_education    0
industry              0
function              0
fraudulent            0
salary_missing        0
salary                0
dtype: int64
```

In [35]:
```
data.head(10)
```

Out[35]:

	job_id	title	location	department	company_profile	
0	1	Marketing Intern	US, NY, New York	Marketing	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, Jam
1	2	Customer Service - Cloud Video Production	NZ, , Auckland	Success	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrai
2	3	Commissioning Machinery Assistant (CMA)	US, IA, Wever		Valor Services provides Workforce Solutions th...	Our client, located in Houstor
3	4	Account Executive - Washington DC	US, DC, Washington	Sales	Our passion for improving quality of life thro...	THE COMPANY: ESRI
4	5	Bill Review Manager	US, FL, Fort Worth		SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Ite Manaş
5	6	Accounting Clerk	US, MD,			Job OverviewApex is ε
6	7	Head of Content (m/f)	DE, BE, Berlin	ANDROIDPIT	Founded in 2009, the Fonpit AG rose with its i...	Your Responsibilities: Mar
7	8	Lead Guest Service Specialist	US, CA, San Francisco		Airenvy's mission is to provide lucrative yet ...	Who is Airenvy?Hey there! V
8	9	HP BSM SME	US, FL, Pensacola		Solutions3 is a woman-owned small business who...	Implementation/Configuration
9	10	Customer Service Associate - Part Time	US, AZ, Phoenix		Novitex Enterprise Solutions, formerly Pitney ...	The Customer Service

Bernouli with smote

In [98]:

```
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report, accuracy_score
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from scipy.sparse import csr_matrix, hstack
from sklearn.feature_extraction.text import CountVectorizer

# Step 1: Combine the text columns into 'combined_text'
text_columns = ['title', 'location', 'department', 'company_profile', 'descri
data['combined_text'] = data[text_columns].apply(lambda x: ' '.join(x), axis=

# Step 2: Vectorize text features using CountVectorizer
vectorizer = CountVectorizer(max_features=5000)
text_features = vectorizer.fit_transform(data['combined_text'])

# Step 3: Separate numeric features
numeric_features = ['telecommuting', 'has_company_logo', 'has_questions']  #
X_numeric = data[numeric_features].values
y = data['fraudulent']

# Step 4: Convert numeric features to sparse matrix for consistency
X_numeric_sparse = csr_matrix(X_numeric)

# Step 5: Combine text features and numeric features
X_combined = hstack([text_features, X_numeric_sparse])

# Step 6: Apply SMOTE to the combined features (text + numeric)
smote = SMOTE(random_state=42)
X_combined_smote, y_smote = smote.fit_resample(X_combined, y)

# Step 7: Split into train and test (ensure you have a proper split before mo
X_train, X_test, y_train, y_test = train_test_split(X_combined_smote, y_smote

# Step 8: Train Exact Bayes (using BernoulliNB) on the resampled data
bernoulli_model = BernoulliNB(alpha=1.0)  # You can adjust alpha
bernoulli_model.fit(X_train, y_train)

# Step 9: Evaluate the model
y_pred = bernoulli_model.predict(X_test)  # Make predictions on the test set

# Step 10: Evaluate accuracy and classification report
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report for Bernoulli Naive Bayes with SMOTE (no PCA):")
print(classification_report(y_test, y_pred))
```

Accuracy: 0.91
Classification Report for Bernoulli Naive Bayes with SMOTE (no PCA):

	precision	recall	f1-score	support
0	0.91	0.90	0.91	5105
1	0.90	0.91	0.91	5104
accuracy			0.91	10209
macro avg	0.91	0.91	0.91	10209
weighted avg	0.91	0.91	0.91	10209

Dialer System for Bernoulli

In [123]:

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import classification_report, accuracy_score, confusion_
import numpy as np

# Train Bernoulli Naive Bayes and generate probabilities
def bernoulli_bayes_probabilities(X_train, y_train, X_test):
    # Initialize and train the Bernoulli Naive Bayes model
    model = BernoulliNB()
    model.fit(X_train, y_train)

    # Get the predicted probabilities for the test data
    probabilities = model.predict_proba(X_test)  # Probability for each class

    return probabilities

# Dialer system to test different thresholds
def dialer_system(y_true, probabilities, thresholds=[0.3, 0.4, 0.5, 0.6, 0.7]
    for threshold in thresholds:
        # Generate predictions based on threshold for fraudulent (class 1)
        y_pred = [1 if prob[1] >= threshold else 0 for prob in probabilities]

        # Display performance metrics
        print(f"\nThreshold: {threshold}")
        print("Accuracy:", accuracy_score(y_true, y_pred))
        print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
        print("Classification Report:\n", classification_report(y_true, y_pre

# Assuming you have already split the data into training and test sets
# X_train, X_test, y_train, y_test

# Generate probabilities with Bernoulli Naive Bayes and apply the dialer syst
probabilities = bernoulli_bayes_probabilities(X_train, y_train, X_test)
dialer_system(y_test, probabilities, thresholds=[0.02,0.01,0.0001, 0.03])  #
```

```
Threshold: 0.02
Accuracy: 0.9053776079929474
Confusion Matrix:
 [[4558  547]
 [ 419 4685]]
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.89      0.90      5105
           1       0.90      0.92      0.91      5104

    accuracy                           0.91     10209
   macro avg       0.91      0.91      0.91     10209
weighted avg       0.91      0.91      0.91     10209


Threshold: 0.01
Accuracy: 0.9051817024194339
Confusion Matrix:
 [[4552  553]
 [ 415 4689]]
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.89      0.90      5105
           1       0.89      0.92      0.91      5104

    accuracy                           0.91     10209
   macro avg       0.91      0.91      0.91     10209
weighted avg       0.91      0.91      0.91     10209


Threshold: 0.0001
Accuracy: 0.905083749632677
Confusion Matrix:
 [[4504  601]
 [ 368 4736]]
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.88      0.90      5105
           1       0.89      0.93      0.91      5104

    accuracy                           0.91     10209
   macro avg       0.91      0.91      0.91     10209
weighted avg       0.91      0.91      0.91     10209


Threshold: 0.03
Accuracy: 0.9048878440591634
Confusion Matrix:
 [[4558  547]
 [ 424 4680]]
Classification Report:
               precision    recall  f1-score   support
```

0	0.91	0.89	0.90	5105
1	0.90	0.92	0.91	5104
accuracy			0.90	10209
macro avg	0.91	0.90	0.90	10209
weighted avg	0.91	0.90	0.90	10209

Exact bayes with smote

In [109]:

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from collections import defaultdict
from sklearn.metrics import classification_report, accuracy_score, confusion_
from imblearn.over_sampling import SMOTE
from scipy.sparse import hstack, csr_matrix
from sklearn.feature_extraction.text import CountVectorizer

# Step 1: Combine the text columns into 'combined_text'
text_columns = ['title', 'location', 'department', 'company_profile', 'descri
data['combined_text'] = data[text_columns].apply(lambda x: ' '.join(x), axis=

# Step 2: Vectorize text features using CountVectorizer
vectorizer = CountVectorizer(max_features=5000)
text_features = vectorizer.fit_transform(data['combined_text'])

# Step 3: Separate numeric features
numeric_features = ['telecommuting', 'has_company_logo', 'has_questions']  #
X_numeric = data[numeric_features].values
y = data['fraudulent']

# Step 4: Convert numeric features to sparse matrix for consistency
X_numeric_sparse = csr_matrix(X_numeric)

# Step 5: Combine text features and numeric features
X_combined = hstack([text_features, X_numeric_sparse])

# Step 6: Apply SMOTE to the combined features (text + numeric)
smote = SMOTE(random_state=42)
X_combined_smote, y_smote = smote.fit_resample(X_combined, y)

# Step 7: Split into train and test (ensure you have a proper split before mo
X_train, X_test, y_train, y_test = train_test_split(X_combined_smote, y_smote

# Step 8: Exact Bayes (Custom model, assuming it's implemented like above)
def exact_bayes_predict(X_train, y_train, X_test):
    classes = np.unique(y_train)
    class_probs = {}
    conditional_probs = defaultdict(lambda: defaultdict(lambda: defaultdict(f

    # Calculate prior probabilities and conditional probabilities
    for c in classes:
        X_class = X_train[y_train == c]
        class_probs[c] = X_class.shape[0] / y_train.shape[0]  # Use .shape[0]

        # Calculate conditional probabilities for each feature
        for col in range(X_train.shape[1]):  # Handle sparse matrix indexing
            col_values = X_class[:, col].toarray().flatten()  # Convert to de
            unique_vals = np.unique(col_values)
            for val in unique_vals:
                conditional_probs[c][col][val] = np.sum(col_values == val) /

    # Make predictions on test data
    predictions = []
    for row in X_test:
        class_scores = {}
```

```python
        row_dense = row.toarray().flatten()  # Convert sparse row to dense
        for c in classes:
            score = np.log(class_probs[c])  # Use log probabilities for numer
            for col in range(X_train.shape[1]):
                val = row_dense[col]
                score += np.log(conditional_probs[c][col].get(val, 1e-6))  #
            class_scores[c] = score
        predictions.append(max(class_scores, key=class_scores.get))
    return predictions

# Step 9: Predict with Exact Bayes model
y_pred_exact_bayes = exact_bayes_predict(X_train, y_train, X_test)

# Step 10: Evaluate the model
accuracy = accuracy_score(y_test, y_pred_exact_bayes)
print(f"Exact Bayes Model Accuracy with SMOTE (no PCA): {accuracy:.2f}")

print("Exact Bayes Model Evaluation with SMOTE (no PCA):")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_exact_bayes))
print("Classification Report:\n", classification_report(y_test, y_pred_exact_
```

```
Exact Bayes Model Accuracy with SMOTE (no PCA): 0.94
Exact Bayes Model Evaluation with SMOTE (no PCA):
Confusion Matrix:
 [[4785  320]
 [ 310 4794]]
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.94      0.94      5105
           1       0.94      0.94      0.94      5104

    accuracy                           0.94     10209
   macro avg       0.94      0.94      0.94     10209
weighted avg       0.94      0.94      0.94     10209
```

Dialer For Exact Bayes

In [180]:

```
import numpy as np
from sklearn.metrics import accuracy_score, confusion_matrix, classification_
from collections import defaultdict

# Dialer system to test different thresholds for Exact Bayes model
def dialer_system_exact_bayes(X_train, y_train, X_test, y_true, thresholds=[0
    # Calculate prior probabilities for each class
    class_probs = {}
    for c in np.unique(y_train):
        class_probs[c] = np.sum(y_train == c) / len(y_train)

    # Calculate conditional probabilities (you have this already in the exact
    conditional_probs = defaultdict(lambda: defaultdict(lambda: defaultdict(f
    for c in np.unique(y_train):
        X_class = X_train[y_train == c]
        for col in range(X_train.shape[1]):  # Handle sparse matrix indexing
            col_values = X_class[:, col].toarray().flatten()  # Convert to de
            unique_vals = np.unique(col_values)
            for val in unique_vals:
                conditional_probs[c][col][val] = np.sum(col_values == val) /

    # Now, let's make predictions and calculate probabilities for each sample
    class_probs_output = []

    for row in X_test:
        row_dense = row.toarray().flatten()  # Convert sparse row to dense
        class_scores = {}
        for c in np.unique(y_train):
            score = np.log(class_probs[c])  # Start with log-prior
            for col in range(X_train.shape[1]):
                val = row_dense[col]
                # Ensure we don't get zero probability
                score += np.log(conditional_probs[c][col].get(val, 1e-6))  #
            class_scores[c] = score

        # Ensure that neither score is NaN or infinite
        if np.isfinite(class_scores[0]) and np.isfinite(class_scores[1]):
            # Softmax-like scaling
            exp_scores = np.exp(np.array([class_scores[0], class_scores[1]]))
            if np.sum(exp_scores) == 0:
                prob_class_1 = 0.5  # If both scores are zero, default to 50%
            else:
                prob_class_1 = exp_scores[1] / np.sum(exp_scores)
        else:
            prob_class_1 = 0.5  # Default to 50% if there's an issue with the

        class_probs_output.append(prob_class_1)

    # Now apply the threshold to the predicted probabilities
    for threshold in thresholds:
        y_pred = [1 if prob >= threshold else 0 for prob in class_probs_outpu

        # Display performance metrics
        print(f"\nThreshold: {threshold}")
        print("Accuracy:", accuracy_score(y_true, y_pred))
        print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
        print("Classification Report:\n", classification_report(y_true, y_pre
```

```
# Example usage (assuming you have already split the data into X_train, X_tes
# Dialer system for Exact Bayes
dialer_system_exact_bayes(X_train, y_train, X_test, y_test, thresholds=[0.5,0
```

```
Threshold: 0.5
Accuracy: 0.7041825839945146
Confusion Matrix:
 [[2169 2936]
 [  84 5020]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.42      0.59      5105
           1       0.63      0.98      0.77      5104

    accuracy                           0.70     10209
   macro avg       0.80      0.70      0.68     10209
weighted avg       0.80      0.70      0.68     10209


Threshold: 0.7
Accuracy: 0.9104711529043001
Confusion Matrix:
 [[4789  316]
 [ 598 4506]]
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.94      0.91      5105
           1       0.93      0.88      0.91      5104

    accuracy                           0.91     10209
   macro avg       0.91      0.91      0.91     10209
weighted avg       0.91      0.91      0.91     10209


Threshold: 0.8
Accuracy: 0.9099813889705162
Confusion Matrix:
 [[4790  315]
 [ 604 4500]]
Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.94      0.91      5105
           1       0.93      0.88      0.91      5104

    accuracy                           0.91     10209
   macro avg       0.91      0.91      0.91     10209
weighted avg       0.91      0.91      0.91     10209


Threshold: 0.9
Accuracy: 0.9108629640513273
Confusion Matrix:
 [[4800  305]
 [ 605 4499]]
Classification Report:
               precision    recall  f1-score   support
```

```
              0       0.89      0.94      0.91      5105
              1       0.94      0.88      0.91      5104

       accuracy                           0.91     10209
      macro avg       0.91      0.91      0.91     10209
   weighted avg       0.91      0.91      0.91     10209


Threshold: 1.0
Accuracy: 0.9077284748751102
Confusion Matrix:
 [[4955  150]
 [ 792 4312]]
Classification Report:
               precision    recall  f1-score   support

              0       0.86      0.97      0.91      5105
              1       0.97      0.84      0.90      5104

       accuracy                           0.91     10209
      macro avg       0.91      0.91      0.91     10209
   weighted avg       0.91      0.91      0.91     10209
```

In [ ]: