```python
# Importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Read the CSV file
df = pd.read_csv("/content/fake_job_postings 123.csv")



# Display the first 5 examples in the dataset
print(df.head())
```

```
   job_id                                          title
location  \
0       1                            Marketing Intern    US, NY, New
York
1       2  Customer Service - Cloud Video Production      NZ, ,
Auckland
2       3    Commissioning Machinery Assistant (CMA)       US, IA,
Wever
3       4          Account Executive - Washington DC  US, DC,
Washington
4       5                         Bill Review Manager  US, FL, Fort
Worth

  department salary_range
company_profile  \
0  Marketing          NaN  We're Food52, and we've created a
groundbreaki...
1    Success          NaN  90 Seconds, the worlds Cloud Video
Production ...
2        NaN          NaN  Valor Services provides Workforce Solutions
th...
3      Sales          NaN  Our passion for improving quality of life
thro...
4        NaN          NaN  SpotSource Solutions LLC is a Global Human
Cap...

                                          description  \
0  Food52, a fast-growing, James Beard Award-winn...
1  Organised - Focused - Vibrant - Awesome!Do you...
2  Our client, located in Houston, is actively se...
```

```
3  THE COMPANY: ESRI – Environmental Systems Rese...
4  JOB TITLE: Itemization Review ManagerLOCATION:...

                                        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...

                                           benefits  telecommuting  \
0                                               NaN              0
1  What you will get from usThrough being part of...              0
2                                               NaN              0
3  Our culture is anything but corporate—we have ...              0
4                            Full Benefits Offered              0

   has_company_logo  has_questions employment_type required_experience
\
0                 1              0           Other          Internship

1                 1              0       Full-time      Not Applicable

2                 1              0             NaN                NaN

3                 1              0       Full-time    Mid-Senior level

4                 1              1       Full-time    Mid-Senior level


  required_education                    industry              function
\
0                NaN                         NaN             Marketing

1                NaN  Marketing and Advertising      Customer Service

2                NaN                         NaN                NaN

3  Bachelor's Degree          Computer Software                Sales

4  Bachelor's Degree     Hospital & Health Care  Health Care Provider


   fraudulent
0           0
1           0
2           0
3           0
4           0
```

```
# Select specific columns
df = df[['title', 'location','company_profile', 'requirements',
'telecommuting', 'has_company_logo', 'has_questions',
'employment_type',
       'required_experience', 'required_education', 'industry',
'function','salary_range',
       'fraudulent']]

# Check for missing values
print(df.isna().apply(pd.value_counts))
print(df.isnull().sum())
```

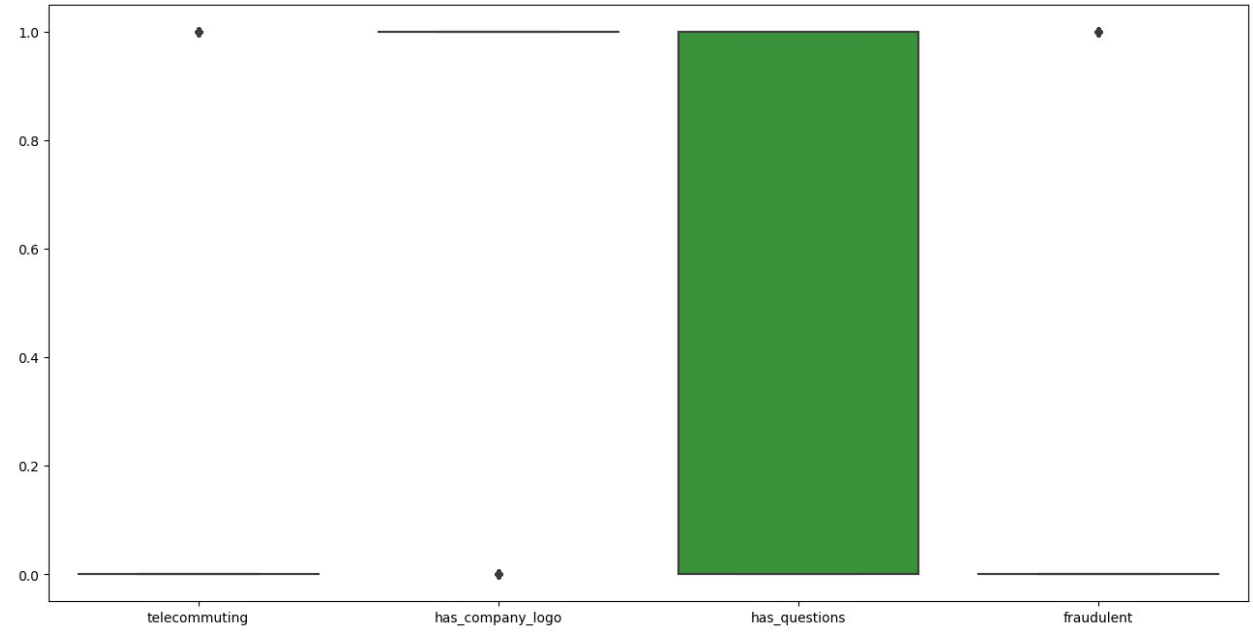
```
         title  location  company_profile  requirements  telecommuting
\
False  17880.0     17534            14572         15185        17880.0

True       NaN       346             3308          2695            NaN


       has_company_logo  has_questions  employment_type
required_experience  \
False           17880.0        17880.0            14409
10830
True                NaN            NaN             3471
7050

       required_education  industry  function  salary_range
fraudulent
False                9775     12977     11425          2868
17880.0
True                 8105      4903      6455         15012
NaN
title                       0
location                  346
company_profile          3308
requirements             2695
telecommuting               0
has_company_logo            0
has_questions               0
employment_type          3471
required_experience      7050
required_education       8105
industry                 4903
function                 6455
salary_range            15012
fraudulent                  0
dtype: int64
```

```
# Remove duplicates
```

```
print(df.duplicated().sum())
df.drop_duplicates(inplace=True)
print(df.duplicated().sum())
```

```
493
0
```

```
# Select numeric and categorical columns
df_num =
df[['telecommuting','has_company_logo','has_questions','fraudulent','s
alary_range']]
df_cat = df[['title', 'location','company_profile',
'requirements','employment_type',
       'required_experience', 'required_education', 'industry',
'function']]

# Visualize numeric data
plt.figure(figsize=[16,8])
sb.boxplot(data=df_num)
plt.show()
```



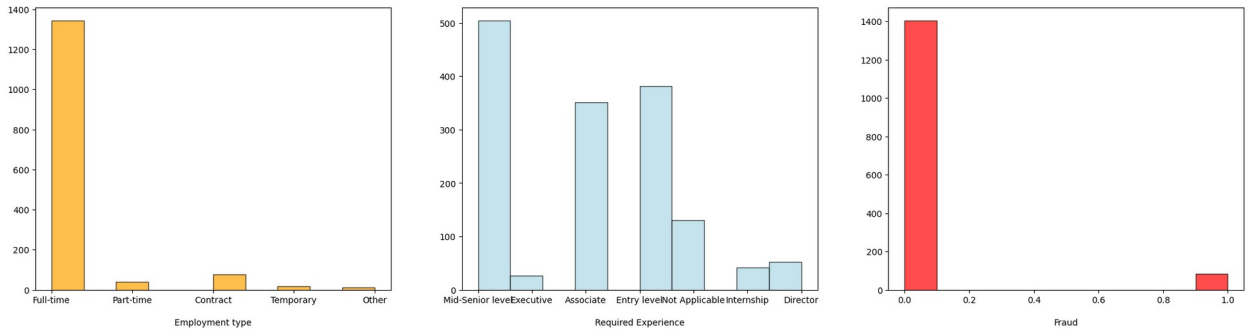

```
# Filter numeric data
df_num = df_num[df_num['telecommuting'] < 0.9 ]
df_num = df_num[df_num['fraudulent'] < 0.9 ]
df_num = df_num[df_num['has_company_logo'] > 0.1 ]

# Handle missing values
print(df.isnull().sum())
```

```
df.dropna(axis=0, how='any', inplace=True)
print(df.isnull().sum())
print(df.shape)
```

```
title                      0
location                 340
company_profile         3255
requirements            2549
telecommuting              0
has_company_logo           0
has_questions              0
employment_type         3397
required_experience     6858
required_education      7889
industry                4769
function                6261
salary_range           14580
fraudulent                 0
dtype: int64
title                  0
location               0
company_profile        0
requirements           0
telecommuting          0
has_company_logo       0
has_questions          0
employment_type        0
required_experience    0
required_education     0
industry               0
function               0
salary_range           0
fraudulent             0
dtype: int64
(1485, 14)
```

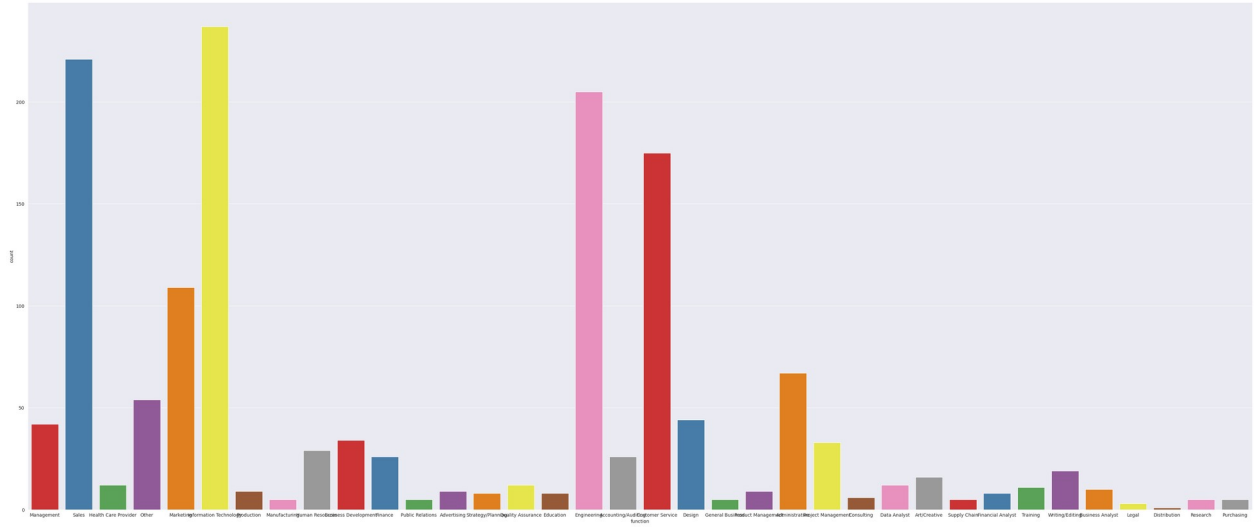
```
# Visualize categorical data
plt.figure(figsize=(25,20))
plt.subplot(3,3,1)
plt.hist(df.employment_type, color='orange', edgecolor='black',
alpha=0.7)
plt.xlabel('\nEmployment type')

plt.subplot(3,3,2)
plt.hist(df.required_experience, color='lightblue', edgecolor='black',
alpha=0.7)
plt.xlabel('\nRequired Experience')

plt.subplot(3,3,3)
```

```
plt.hist(df.fraudulent, color='red', edgecolor='black', alpha=0.7)
plt.xlabel('\nFraud')

plt.show()
```




```
# Additional visualization
plt.figure(figsize=(48,20))
sb.set_style("darkgrid")
sb.countplot(x='function', data=df, palette='Set1')
df_jobs = df[(df['employment_type'] == 'Full-time') &
(df['fraudulent']== 0)]
print(df_jobs.shape)
```

```
(1269, 14)
```



```
# Checking the counts of each unique value in 'title'
print(df_jobs['title'].value_counts())
```

```
Agent-Inbound Sales Position                                    12
Marketing Manager                                                8
```

```
Customer Service Representative                                    8
Account Manager                                                    7
Technical Support Associate                                        7
                                                                  ..
Sales Executive for Content Marketing Firm                         1
Wait Staff Position Available                                      1
Graduate application - HW Design                                   1
Senior Quant Analyst                                               1
Portfolio Development Associate - Paris & rest of France           1
Name: title, Length: 1027, dtype: int64


# Maximum count in 'title'
print(df_jobs['title'].value_counts().max())


12

# Displaying the first row of the DataFrame
print(df.head(1))


                   title        location  \
6  Head of Content (m/f)  DE, BE, Berlin

                                      company_profile  \
6  Founded in 2009, the Fonpit AG rose with its i...

                                         requirements  telecommuting  \
6  Your Know-How:                                ...              0

   has_company_logo  has_questions employment_type required_experience
\
6                 1              1       Full-time    Mid-Senior level


  required_education      industry    function salary_range
fraudulent
6    Master's Degree  Online Media  Management  20000-28000
0

# Filtering the DataFrame for fraudulent entries
df_industry = df[df['fraudulent'] == 1]
print(df_industry.shape)


(82, 14)

# Checking counts of unique values in the 'industry' column for
fraudulent entries
print(df_industry['industry'].value_counts())
```

```
Oil & Energy                                19
Real Estate                                 10
Consumer Services                            8
Financial Services                           6
Computer & Network Security                  5
Retail                                       4
Management Consulting                        3
Hospital & Health Care                       2
Accounting                                   2
Telecommunications                           2
Hospitality                                  2
Human Resources                              2
Biotechnology                                2
Insurance                                    2
Transportation/Trucking/Railroad             2
Construction                                 1
Defense & Space                              1
Computer Networking                          1
Mechanical or Industrial Engineering         1
Warehousing                                  1
Media Production                             1
Information Services                         1
Logistics and Supply Chain                   1
Security and Investigations                  1
Military                                     1
Electrical/Electronic Manufacturing          1
Name: industry, dtype: int64
```

```
# Checking counts of 'fraudulent' values
print(df['fraudulent'].value_counts())
```

```
0    1403
1      82
Name: fraudulent, dtype: int64
```

```
# Extracting values from the 'fraudulent' column
print(df['fraudulent'].values)
```

```
[0 0 0 ... 1 0 0]
```

```
# Separating fraudulent and non-fraudulent entries
fraud = df[df['fraudulent'] == 1]
print(fraud.shape)

not_fraud = df[df['fraudulent'] == 0]
print(not_fraud.shape)
```

```
(82, 14)
(1403, 14)
```

```
# Balancing the dataset by sampling the same number of fraudulent
entries as non-fraudulent ones
fraud = fraud.sample(1403, replace=True)
print(fraud.shape, not_fraud.shape)
```

```
(1403, 14) (1403, 14)
```

```
# Combining fraudulent and non-fraudulent entries into a new DataFrame
df = fraud.append(not_fraud)
df.reset_index(inplace=True)
```

```
<ipython-input-22-bda44e33ab32>:2: FutureWarning: The frame.append
method is deprecated and will be removed from pandas in a future
version. Use pandas.concat instead.
  df = fraud.append(not_fraud)
```

```
# Encoding categorical features using LabelEncoder
le = LabelEncoder()
df['title'] = le.fit_transform(df['title'])
df['location'] = le.fit_transform(df['location'])
df['company_profile'] = le.fit_transform(df['company_profile'])
df['requirements'] = le.fit_transform(df['requirements'])
df['employment_type'] = le.fit_transform(df['employment_type'])
df['required_experience'] =
le.fit_transform(df['required_experience'])
df['required_education'] = le.fit_transform(df['required_education'])
df['industry'] = le.fit_transform(df['industry'])
df['function'] = le.fit_transform(df['function'])
df['salary_range'] = le.fit_transform(df['salary_range'])
```

```
# Splitting the data into features (X) and target variable (Y)
X = df[['index', 'title', 'location', 'company_profile',
'requirements',
        'telecommuting', 'has_company_logo', 'has_questions',
'employment_type',
        'required_experience', 'required_education', 'industry',
'function',
        'salary_range']].values
Y = df[['fraudulent']].values

# Splitting the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y)
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(2104, 14) (702, 14) (2104, 1) (702, 1)
```

```
# Logistic Regression
LgR = LogisticRegression()
LgR.fit(X_train, Y_train)
Y_pred_lgr = LgR.predict(X_test)
Y_test_lgr = Y_test.flatten()
Y_pred_lgr = Y_pred_lgr.flatten()
print(Y_test_lgr.shape, Y_pred_lgr.shape)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/
validation.py:1143: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

(702,) (702,)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
# Creating a DataFrame for Logistic Regression results
df_lgr = pd.DataFrame({'Y_test': Y_test_lgr, 'Y_pred': Y_pred_lgr})
print(df_lgr)

# Calculating and printing accuracy for Logistic Regression
accuracy_lgr = accuracy_score(Y_pred_lgr, Y_test_lgr)
print(f"Logistic Regression Accuracy: {accuracy_lgr}")
```

```
     Y_test  Y_pred
0         1       1
1         1       1
2         1       1
3         0       1
4         0       0
..      ...     ...
697       1       1
698       1       1
699       0       1
700       0       0
701       0       0
```

```
[702 rows x 2 columns]
Logistic Regression Accuracy: 0.6994301994301995
```

```
# K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
Y_pred_knn = knn.predict(X_test)
Y_test_knn = Y_test.flatten()
Y_pred_knn = Y_pred_knn.flatten()
print(Y_test_knn.shape, Y_pred_knn.shape)

# Creating a DataFrame for KNN results
df_knn = pd.DataFrame({'Y_test': Y_test_knn, 'Y_pred': Y_pred_knn})
print(df_knn)

# Calculating and printing accuracy for KNN
accuracy_knn = accuracy_score(Y_pred_knn, Y_test_knn)
print(f"KNN Accuracy: {accuracy_knn}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/
_classification.py:215: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  return self._fit(X, y)

(702,) (702,)
     Y_test  Y_pred
0         1       1
1         1       1
2         1       1
3         0       0
4         0       1
..      ...     ...
697       1       1
698       1       1
699       0       0
700       0       0
701       0       0

[702 rows x 2 columns]
KNN Accuracy: 0.9259259259259259
```

```
# Random Forest Classifier
rfc = RandomForestClassifier(n_estimators=5)
rfc.fit(X_train, Y_train)
Y_pred_rfc = rfc.predict(X_test)
Y_test_rfc = Y_test.flatten()
Y_pred_rfc = Y_pred_rfc.flatten()
```

```python
print(Y_test_rfc.shape, Y_pred_rfc.shape)

# Creating a DataFrame for Random Forest Classifier results
df_rfc = pd.DataFrame({'Y_test': Y_test_rfc, 'Y_pred': Y_pred_rfc})
print(df_rfc)

# Calculating and printing accuracy for Random Forest Classifier
accuracy_rfc = accuracy_score(Y_pred_rfc, Y_test_rfc)
print(f"Random Forest Classifier Accuracy: {accuracy_rfc}")
```

```
(702,) (702,)
     Y_test  Y_pred
0         1       1
1         1       1
2         1       1
3         0       0
4         0       0
..      ...     ...
697       1       1
698       1       1
699       0       0
700       0       0
701       0       0

[702 rows x 2 columns]
Random Forest Classifier Accuracy: 0.9943019943019943

<ipython-input-28-a0afb7cfb637>:3: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
  rfc.fit(X_train, Y_train)
```

```python
from sklearn.metrics import precision_score, recall_score, f1_score

# For Logistic Regression
precision_lgr = precision_score(Y_test_lgr, Y_pred_lgr)
recall_lgr = recall_score(Y_test_lgr, Y_pred_lgr)
f1_lgr = f1_score(Y_test_lgr, Y_pred_lgr)

# For K-Nearest Neighbors
precision_knn = precision_score(Y_test_knn, Y_pred_knn)
recall_knn = recall_score(Y_test_knn, Y_pred_knn)
f1_knn = f1_score(Y_test_knn, Y_pred_knn)

# For Random Forest Classifier
precision_rfc = precision_score(Y_test_rfc, Y_pred_rfc)
recall_rfc = recall_score(Y_test_rfc, Y_pred_rfc)
f1_rfc = f1_score(Y_test_rfc, Y_pred_rfc)

# Print the scores for each model
print(f"Logistic Regression Precision: {precision_lgr}, Recall:
```

```
{recall_lgr}, F1: {f1_lgr}")
print(f"K-Nearest Neighbors Precision: {precision_knn}, Recall:
{recall_knn}, F1: {f1_knn}")
print(f"Random Forest Precision: {precision_rfc}, Recall:
{recall_rfc}, F1: {f1_rfc}")
```

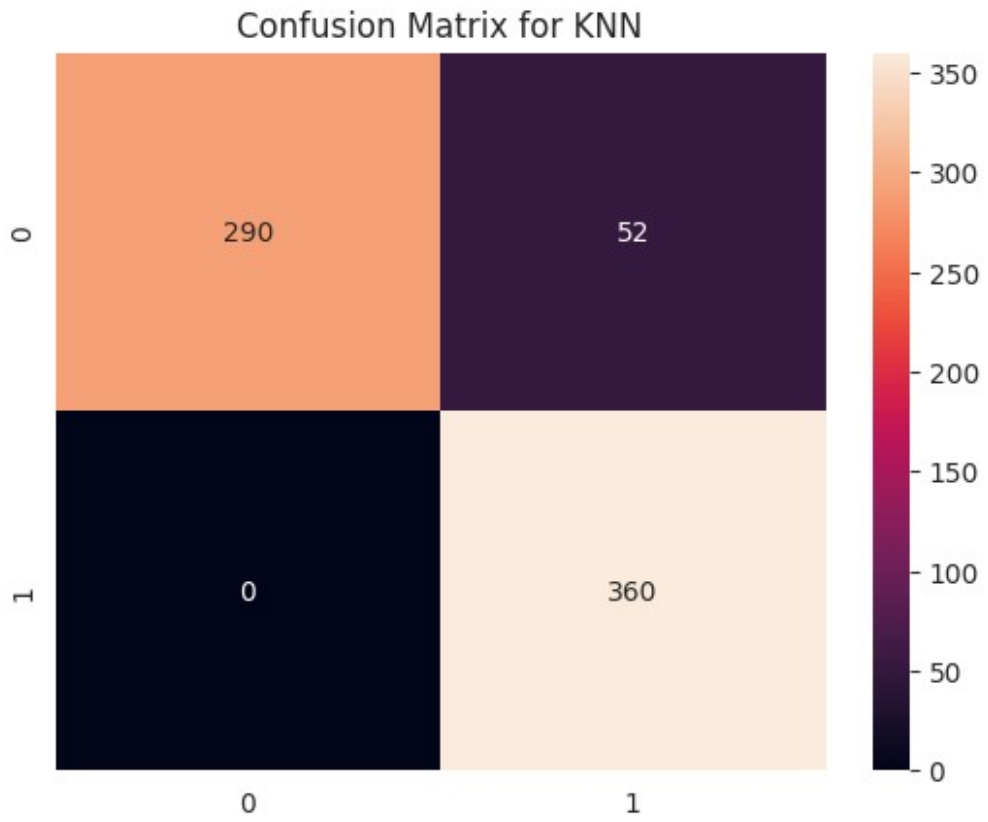
```
Logistic Regression Precision: 0.6935064935064935, Recall:
0.7416666666666667, F1: 0.7167785234899328
K-Nearest Neighbors Precision: 0.8737864077669902, Recall: 1.0, F1:
0.932642487046632
Random Forest Precision: 0.989010989010989, Recall: 1.0, F1:
0.994475138121547
```

```
from sklearn.metrics import confusion_matrix

y_pred_lgr = LgR.predict(X_test)  # Logistic Regression predictions
y_pred_knn = knn.predict(X_test)  # K-Nearest Neighbors predictions
y_pred_rfc = rfc.predict(X_test)  # Random Forest predictions

cm_lgr = confusion_matrix(Y_test, y_pred_lgr)
cm_knn = confusion_matrix(Y_test, y_pred_knn)
cm_rfc = confusion_matrix(Y_test, y_pred_rfc)

import seaborn as sns
ax = sns.heatmap(cm_lgr, annot=True, fmt="d")
ax.set_title('Confusion Matrix for Logistic Regression')
```

```
Text(0.5, 1.0, 'Confusion Matrix for Logistic Regression')
```

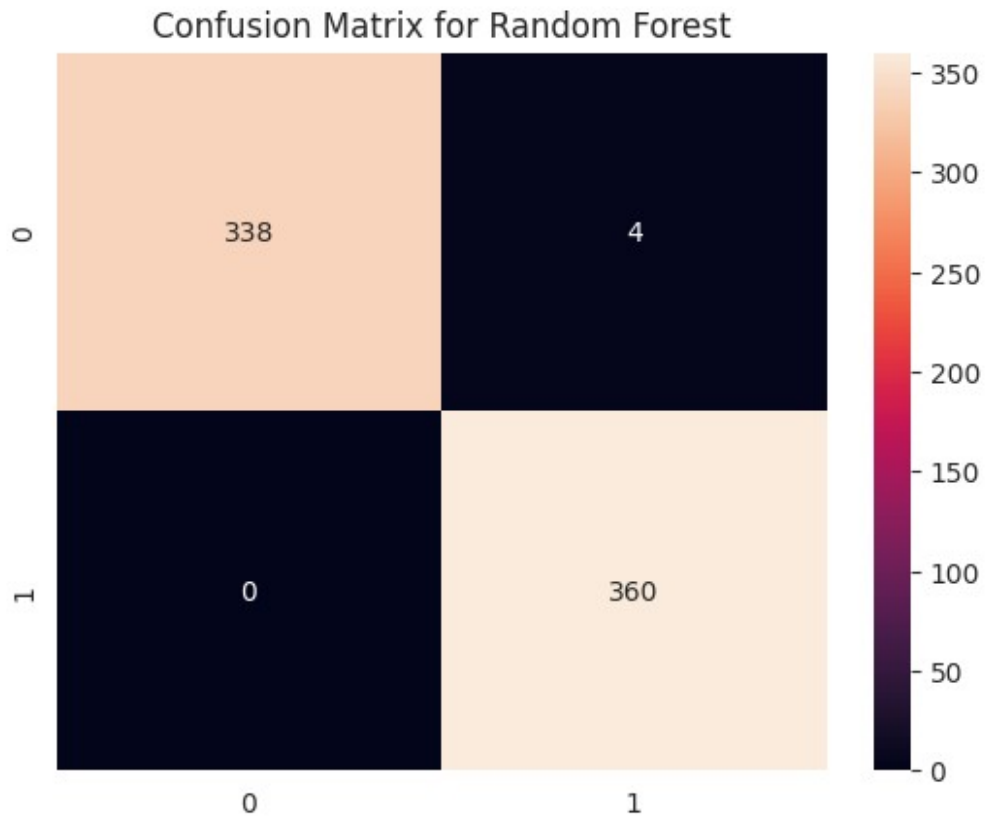

```
ax = sns.heatmap(cm_knn, annot=True, fmt="d")
ax.set_title('Confusion Matrix for KNN')
```

```
Text(0.5, 1.0, 'Confusion Matrix for KNN')
```




```
ax = sns.heatmap(cm_rfc, annot=True, fmt="d")
ax.set_title('Confusion Matrix for Random Forest')
```

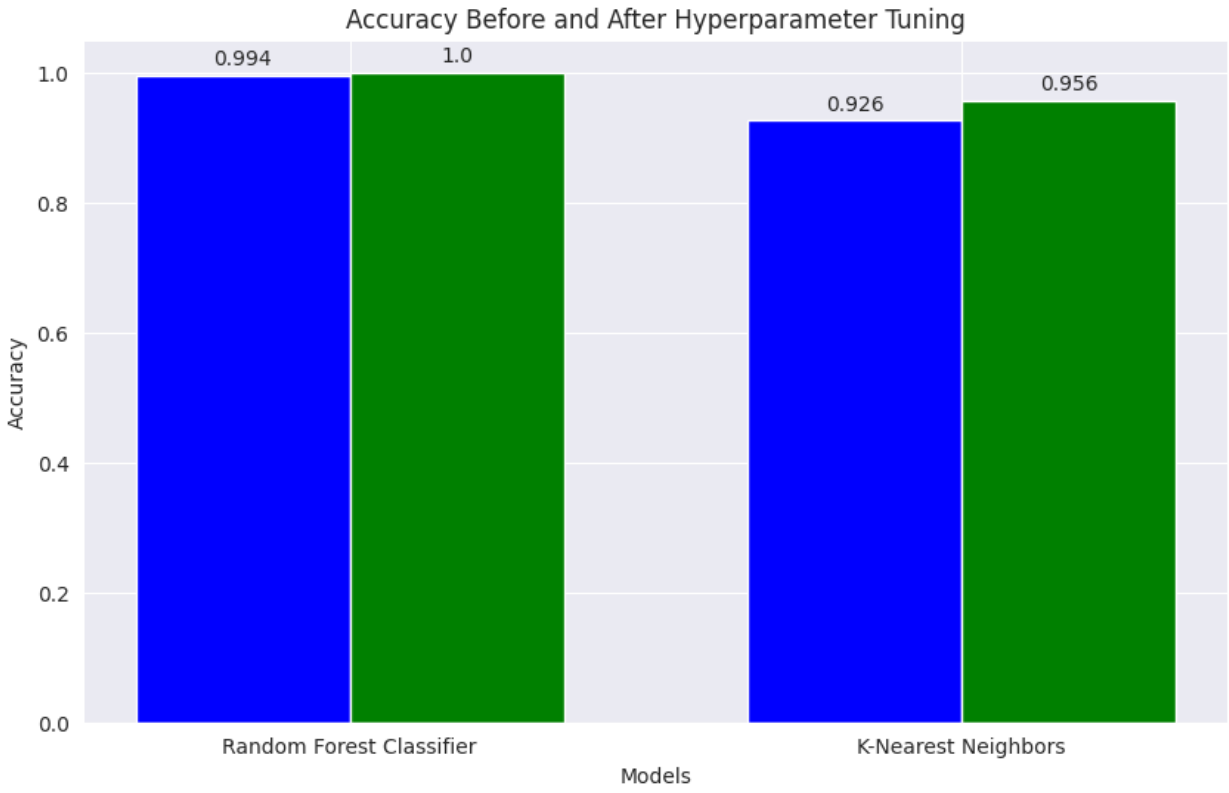
```
Text(0.5, 1.0, 'Confusion Matrix for Random Forest')
```




```
from sklearn.model_selection import GridSearchCV


# Define hyperparameter grid for Random Forest Classifier
param_grid_rfc = {'n_estimators': [10, 50, 100, 200],
                  'max_depth': [None, 10, 20, 30],
                  'min_samples_split': [2, 5, 10],
                  'min_samples_leaf': [1, 2, 4]}

# Initialize Random Forest Classifier model
rfc = RandomForestClassifier()

# Create GridSearchCV object for Random Forest Classifier
grid_search_rfc = GridSearchCV(estimator=rfc,
param_grid=param_grid_rfc, cv=5, scoring='accuracy')

# Fit the data to find the best hyperparameters
grid_search_rfc.fit(X_train, Y_train)

# Display the best hyperparameters
print("Best Hyperparameters for Random Forest Classifier:",
grid_search_rfc.best_params_)

# Get the best model
best_rfc = grid_search_rfc.best_estimator_
```

```
# Use the best model for prediction
Y_pred_rfc_grid = best_rfc.predict(X_test)

# Calculate and print accuracy for Random Forest Classifier with Grid
Search
accuracy_rfc_grid = accuracy_score(Y_pred_rfc_grid, Y_test.flatten())
print(f"Random Forest Classifier Accuracy (Grid Search):
{accuracy_rfc_grid}")

from sklearn.neighbors import KNeighborsClassifier

# Define hyperparameter grid for K-Nearest Neighbors
param_grid_knn = {'n_neighbors': [3, 5, 7, 9],
                  'weights': ['uniform', 'distance'],
                  'p': [1, 2]}

# Initialize K-Nearest Neighbors model
knn = KNeighborsClassifier()

# Create GridSearchCV object for K-Nearest Neighbors
grid_search_knn = GridSearchCV(estimator=knn,
param_grid=param_grid_knn, cv=5, scoring='accuracy')

# Fit the data to find the best hyperparameters
grid_search_knn.fit(X_train, Y_train)

# Display the best hyperparameters for K-Nearest Neighbors
print("Best Hyperparameters for K-Nearest Neighbors:",
grid_search_knn.best_params_)

# Get the best model for K-Nearest Neighbors
best_knn = grid_search_knn.best_estimator_

# Use the best model for prediction
Y_pred_knn_grid = best_knn.predict(X_test)

# Calculate and print accuracy for K-Nearest Neighbors with Grid
Search
accuracy_knn_grid = accuracy_score(Y_pred_knn_grid, Y_test.flatten())
print(f"K-Nearest Neighbors Accuracy (Grid Search):
{accuracy_knn_grid}")

# Accuracy before and after tuning for K-Nearest Neighbors
accuracy_before_tuning_knn = accuracy_knn
accuracy_after_tuning_knn = accuracy_knn_grid
model_names_knn = ['K-Nearest Neighbors']

import matplotlib.pyplot as plt
import numpy as np
```

```
# Function to plot bar graph
def plot_accuracy_comparison(before_tuning, after_tuning,
model_names):
    fig, ax = plt.subplots(figsize=(10, 6))

    bar_width = 0.35
    index = np.arange(len(model_names))

    # Bar plots for accuracy before tuning
    bars_before = plt.bar(index, before_tuning, bar_width,
label='Before Tuning', color='b')

    # Bar plots for accuracy after tuning
    bars_after = plt.bar(index + bar_width, after_tuning, bar_width,
label='After Tuning', color='g')

    # Add labels, title, and legend
    plt.xlabel('Models')
    plt.ylabel('Accuracy')
    plt.title('Accuracy Before and After Hyperparameter Tuning')
    plt.xticks(index + bar_width / 2, model_names)
    #plt.legend()

    # Display accuracy values on top of the bars
    for bar in bars_before + bars_after:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2, yval + 0.01,
round(yval, 3), ha='center', va='bottom')

    plt.show()
    # Accuracy before and after tuning for Logistic Regression and
Random Forest Classifier
accuracy_before_tuning = [accuracy_rfc]
accuracy_after_tuning = [accuracy_rfc_grid]
model_names = ['Random Forest Classifier']

# Plot the bar graph
plot_accuracy_comparison(accuracy_before_tuning +
[accuracy_before_tuning_knn],
                         accuracy_after_tuning +
[accuracy_after_tuning_knn],
                         model_names + model_names_knn)
```


Accuracy Before and After Hyperparameter Tuning
0.994
1.0
0.926
0.956
1.0
0.8
0.6
0.4
0.2
0.0
Accuracy
Random Forest Classifier
K-Nearest Neighbors
Models