In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_val_score, StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve
```

In [2]:
```
# We read the CSV file into a DataFrame so we can work with it in Python
df = pd.read_csv(r"C:\Users\Top Prix\Downloads\archive (2)\clicks_dataset.csv", encoding='ISO-8859-1')

# 3️⃣ BASIC EXPLORATION (EDA = Exploratory Data Analysis)
# Look at first rows, data types, and summary stats
print(df.head())      # quick preview of first 5 rows
print(df.info())      # data types and missing values
print(df.describe())  # basic statistics for numeric columns
```

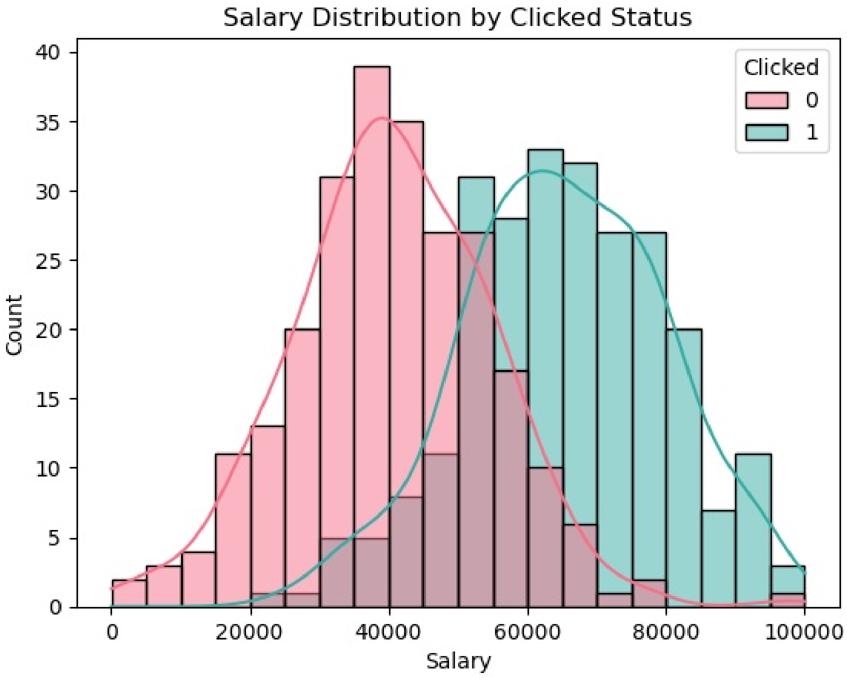
```
             Names                                             emails  \
0    Martina Avila  cubilia.Curae.Phasellus@quisaccumsanconvallis.edu
1    Harlan Barnes                                eu.dolor@diam.co.uk
2  Naomi Rodriquez  vulputate.mauris.sagittis@ametconsectetueradip...
3  Jade Cunningham                            malesuada@dignissim.com
4     Cedric Leach     felis.ullamcorper.viverra@egetmollislectus.net

        Country  Time Spent on Site       Salary  Clicked
0      Bulgaria           25.649648  55330.06006        0
1        Belize           32.456107  79049.07674        1
2       Algeria           20.945978  41098.60826        0
3  Cook Islands           54.039325  37143.35536        1
4        Brazil           34.249729  37355.11276        0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 6 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Names               499 non-null    object
 1   emails              499 non-null    object
 2   Country             499 non-null    object
 3   Time Spent on Site  499 non-null    float64
 4   Salary              499 non-null    float64
 5   Clicked             499 non-null    int64
dtypes: float64(2), int64(1), object(3)
memory usage: 23.5+ KB
None
       Time Spent on Site         Salary     Clicked
count          499.000000     499.000000  499.000000
mean            32.920178   52896.992469    0.501002
std              9.103455   18989.183150    0.500501
min              5.000000      20.000000    0.000000
25%             26.425044   38888.117260    0.000000
50%             33.196067   52840.913110    1.000000
75%             39.114995   65837.288190    1.000000
max             60.000000  100000.000000    1.000000
```

In [3]:
```
# --- Class balance plot ---
sns.countplot(x='Clicked', data=df, palette='husl')
plt.title("Class Distribution (Clicked vs Not Clicked)")
plt.show()
```

```
C:\Users\Top Prix\AppData\Local\Temp\ipykernel_9732\1994828306.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Clicked', data=df, palette='husl')
```

In [4]:
```
# --- Scatter plot to see if time & salary relate to clicking ---
sns.scatterplot(x='Time Spent on Site', y='Salary', hue='Clicked', data=df, palette='husl')
plt.title("Time Spent on Site vs Salary")
plt.show()
```

In [5]:
```
# --- Salary distribution by click status ---
sns.histplot(df, x='Salary', hue='Clicked', bins=20, palette='husl', kde=True)
plt.title("Salary Distribution by Clicked Status")
plt.show()
```



In [6]:
```
# 4 FEATURE ENGINEERING
# Here we create new columns that may help the model
df['Time_x_Salary'] = df['Time Spent on Site'] * df['Salary']          # interaction term
df['Log_Salary'] = np.log1p(df['Salary'])                              # log transform to reduce skew
df['Salary_Level'] = pd.cut(df['Salary'],                              # bucket salary into categories
                            bins=[0, 30000, 60000, 100000],
                            labels=['Low', 'Medium', 'High'])

# Convert categorical columns to 0/1 (dummy variables)
df = pd.get_dummies(df, columns=['Country', 'Salary_Level'], drop_first=True)

# Remove irrelevant columns
df = df.drop(columns=['emails', 'Names'])
```

In [7]:
```
df
```

Out[7]:

	Time Spent on Site	Salary	Clicked	Time_x_Salary	Log_Salary	Country_Algeria	Country_American Samoa	Country_Andorra	Country_A
0	25.649648	55330.06006	0	1.419197e+06	10.921090	False	False	False	
1	32.456107	79049.07674	1	2.565625e+06	11.277837	False	False	False	
2	20.945978	41098.60826	0	8.608505e+05	10.623754	True	False	False	
3	54.039325	37143.35536	1	2.007202e+06	10.522567	False	False	False	
4	34.249729	37355.11276	0	1.279402e+06	10.528252	False	False	False	
...	...	...	...	...	...	...	...	...	
494	19.222746	44969.13495	0	8.644303e+05	10.713754	False	False	False	
495	22.665662	41686.20425	0	9.448454e+05	10.637950	False	False	False	
496	35.320239	23989.80864	0	8.473258e+05	10.085426	False	False	False	
497	26.539170	31708.57054	0	8.415192e+05	10.364374	False	False	False	
498	32.386148	74331.35442	1	2.407306e+06	11.216302	False	False	False	

499 rows × 216 columns

In [8]:
```
#  SPLIT INTO TRAIN AND TEST SETS

# stratify=y ensures both sets have same click/not click ratio
X = df.drop('Clicked', axis=1)
y = df['Clicked']
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [9]:
```
# SET UP MODELS FOR CROSS-VALIDATION

log_model = Pipeline([
    ("scaler", StandardScaler()),
```

```
    ("clf", LogisticRegression(max_iter=500, random_state=42))
])
```

In [10]:
```
# Random Forest doesn't need scaling
rf_model = RandomForestClassifier(random_state=42, n_jobs=-1)
```

In [11]:
```
# Cross-validation setup: 5 folds, keep class balance
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

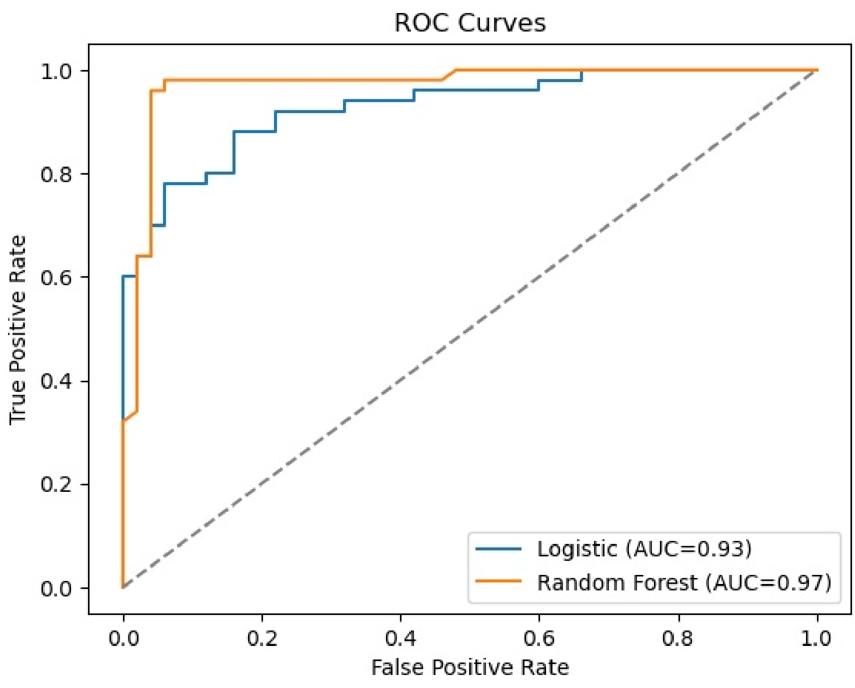
In [12]:
```
# Run cross-validation on training set
cv_log_acc = cross_val_score(log_model, X_train, y_train, cv=cv, scoring="accuracy").mean()
cv_log_auc = cross_val_score(log_model, X_train, y_train, cv=cv, scoring="roc_auc").mean()

cv_rf_acc = cross_val_score(rf_model, X_train, y_train, cv=cv, scoring="accuracy").mean()
cv_rf_auc = cross_val_score(rf_model, X_train, y_train, cv=cv, scoring="roc_auc").mean()

print(f"Logistic Regression CV  Acc: {cv_log_acc:.3f} | AUC: {cv_log_auc:.3f}")
print(f"Random Forest       CV  Acc: {cv_rf_acc:.3f} | AUC: {cv_rf_auc:.3f}")
```

```
Logistic Regression CV  Acc: 0.845 | AUC: 0.920
Random Forest       CV  Acc: 0.882 | AUC: 0.948
```

In [13]:
```
#  FINAL TRAINING AND TEST EVALUATION
# Fit both models on the full training data
log_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

# Predict on test set and evaluate
log_acc = accuracy_score(y_test, log_model.predict(X_test))
rf_acc  = accuracy_score(y_test, rf_model.predict(X_test))

log_auc = roc_auc_score(y_test, log_model.predict_proba(X_test)[:, 1])
rf_auc  = roc_auc_score(y_test, rf_model.predict_proba(X_test)[:, 1])

print("\n=== Test Set Performance ===")
print(f"Logistic Regression - Accuracy: {log_acc:.3f}, AUC: {log_auc:.3f}")
print(f"Random Forest       - Accuracy: {rf_acc:.3f}, AUC: {rf_auc:.3f}")
```

```
=== Test Set Performance ===
Logistic Regression - Accuracy: 0.840, AUC: 0.928
Random Forest       - Accuracy: 0.950, AUC: 0.970
```

In [14]:
```
#  VISUALIZE ROC CURVES FOR BOTH MODELS
fpr_log, tpr_log, _ = roc_curve(y_test, log_model.predict_proba(X_test)[:, 1])
fpr_rf,  tpr_rf,  _ = roc_curve(y_test, rf_model.predict_proba(X_test)[:, 1])

plt.plot(fpr_log, tpr_log, label=f"Logistic (AUC={log_auc:.2f})")
plt.plot(fpr_rf,  tpr_rf,  label=f"Random Forest (AUC={rf_auc:.2f})")
plt.plot([0, 1], [0, 1], linestyle="--", color="grey")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curves")
plt.legend()
plt.show()
```



In [15]:
```
from sklearn.metrics import confusion_matrix
```

```
corr_matrix=confusion_matrix(y_test, log_model.predict(X_test))
sns.heatmap(corr_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Clicked', 'Clicked'],
            yticklabels=['Not Clicked', 'Clicked'])
```

Out[15]: <Axes: >

In [16]:

```
corr_matrix1= confusion_matrix(y_test, rf_model.predict(X_test))
sns.heatmap(corr_matrix1, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Not Clicked', 'Clicked'],
            yticklabels=['Not Clicked', 'Clicked'])
```

Out[16]: <Axes: >