

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, f_classif, RFE
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive")

```
df = pd.read_csv('/content/drive/MyDrive/AI CODE/healthcare_fraud_detection.csv')
df
```

	Provider_ID	Claim_ID	Patient_Age	Patient_Gender	Diagnosis_Code	Procedure_Code	Claim_Status
0	P0052	C0000000	37	Male	I25.10	36415	0
1	P0121	C0000001	21	Female	E11.9	99213	0
2	P0140	C0000002	78	Female	J06.9	93000	0
3	P0202	C0000003	65	Male	I10	93000	0
4	P0135	C0000004	36	Male	M54.5	85025	0
...	...	...	...	...	...	...	...
9995	P0289	C0009995	79	Male	E78.5	71046	0
9996	P0248	C0009996	66	Male	K21.9	93000	0
9997	P0122	C0009997	43	Female	F41.9	99213	0
9998	P0072	C0009998	39	Female	I10	93000	0
9999	P0137	C0009999	26	Male	J18.9	71046	0

10000 rows × 8 columns

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
num_cols = df.select_dtypes(include=['float64', 'int64']).columns
cat_cols = df.select_dtypes(include=['object']).columns
```

```
num_imputer = SimpleImputer(strategy='median')
```

```
df[num_cols] = num_imputer.fit_transform(df[num_cols])
```

```
cat_imputer = SimpleImputer(strategy='most_frequent')
df[cat_cols] = cat_imputer.fit_transform(df[cat_cols])
```

```
le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col].astype(str))
```

```
X = df.drop(columns=['Is_Fraud', 'Provider_ID', 'Claim_ID', 'Claim_Submission_Date'])
y = df['Is_Fraud']
```

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

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
rf = RandomForestClassifier(random_state=42)
```

```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}
```

```
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=3, scoring='accuracy', n_
grid_search.fit(X_train_scaled, y_train)
```

```
best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(X_test_scaled)
```

```
print(f"أفضل الإعدادات المكتشفة: {grid_search.best_params_}")
print("\nتقرير أداء النموذج المطور:")
print(classification_report(y_test, y_pred))
```

أفضل الإعدادات المكتشفة: {'max_depth': 20, 'min_samples_split': 2, 'n_estimators': 50}

تقرير أداء النموذج المطور:

	precision	recall	f1-score	support
0.0	0.98	0.99	0.99	1847
1.0	0.92	0.79	0.85	153
accuracy			0.98	2000
macro avg	0.95	0.89	0.92	2000
weighted avg	0.98	0.98	0.98	2000

```
selector_kbest = SelectKBest(score_func=f_classif, k=5)
selector_kbest.fit(X_train_scaled, y_train)
```

```
cols_kbest = X.columns[selector_kbest.get_support()]
```

```
logreg = LogisticRegression()
selector_rfe = RFE(estimator=logreg, n_features_to_select=5)
selector_rfe.fit(X_train_scaled, y_train)
cols_rfe = X.columns[selector_rfe.get_support()]

print(f"الميزات المختارة (SelectKBest): {list(cols_kbest)}")
print(f"الميزات المختارة (RFE): {list(cols_rfe)}")
```

```
الميزات المختارة (SelectKBest): ['Claim_Amount', 'Approved_Amount', 'Days_Between_Service_and_Claim']
الميزات المختارة (RFE): ['Claim_Amount', 'Approved_Amount', 'Days_Between_Service_and_Claim', 'Numb
```

```
def evaluate_on_features(feature_list, title):
    print(f"\n--- تقييم النموذج باستخدام ميزات {title} ---")

    X_train_sel = X_train[feature_list]
    X_test_sel = X_test[feature_list]

    scaler_sel = StandardScaler()
    X_train_sel_scaled = scaler_sel.fit_transform(X_train_sel)
    X_test_sel_scaled = scaler_sel.transform(X_test_sel)

    best_rf.fit(X_train_sel_scaled, y_train)
    y_pred_sel = best_rf.predict(X_test_sel_scaled)

    print(f"الدقة (Accuracy): {accuracy_score(y_test, y_pred_sel):.4f}")
    print(classification_report(y_test, y_pred_sel))

evaluate_on_features(cols_kbest, "SelectKBest")
evaluate_on_features(cols_rfe, "RFE")
```

```
--- تقييم النموذج باستخدام ميزات SelectKBest ---
الدقة (Accuracy): 0.9885
```

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	1847
1.0	0.96	0.89	0.92	153
accuracy			0.99	2000
macro avg	0.97	0.94	0.96	2000
weighted avg	0.99	0.99	0.99	2000

```
--- تقييم النموذج باستخدام ميزات RFE ---
الدقة (Accuracy): 0.9885
```

	precision	recall	f1-score	support
0.0	0.99	1.00	0.99	1847
1.0	0.96	0.89	0.92	153
accuracy			0.99	2000
macro avg	0.97	0.94	0.96	2000
weighted avg	0.99	0.99	0.99	2000

