

```

import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.feature_selection import SelectKBest, mutual_info_classif, RFE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv('/content/Dry_Eye_Dataset.csv')

bp = df['Blood pressure'].astype(str).str.split('/', expand=True)
df['Systolic_BP'] = pd.to_numeric(bp[0], errors='coerce')
df['Diastolic_BP'] = pd.to_numeric(bp[1], errors='coerce')
df = df.drop(columns=['Blood pressure'])

le = LabelEncoder()
df['Dry Eye Disease'] = le.fit_transform(df['Dry Eye Disease'])

X = df.drop(columns=['Dry Eye Disease'])
y = df['Dry Eye Disease']

categorical_cols = X.select_dtypes(include='object').columns.tolist()
numeric_cols = X.select_dtypes(exclude='object').columns.tolist()

numeric_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer([
    ('num', numeric_transformer, numeric_cols),
    ('cat', categorical_transformer, categorical_cols)
])

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

```

```

models = {
    'Logistic Regression': {
        'model': LogisticRegression(max_iter=1000),
        'params': {
            'classifier__C': [0.1, 1, 10]
        }
    },
    'Random Forest': {
        'model': RandomForestClassifier(random_state=42),
        'params': {
            'classifier__n_estimators': [100, 200],
            'classifier__max_depth': [None, 10, 20],
            'classifier__min_samples_split': [2, 5]
        }
    }
}

```

```

    }
}

results_before_fs = []

for name, config in models.items():
    pipe = Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', config['model'])
    ])

    grid = GridSearchCV(
        pipe,
        config['params'],
        cv=5,
        scoring='accuracy',
        n_jobs=-1
    )

    grid.fit(X_train, y_train)
    y_pred = grid.predict(X_test)

    results_before_fs.append({
        'Model': name,
        'Best Params': grid.best_params_,
        'Accuracy': accuracy_score(y_test, y_pred)
    })

print(f'\n{name} Before Feature Selection')
print('Best Params:', grid.best_params_)
print('Accuracy:', accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

```

Logistic Regression Before Feature Selection

Best Params: {'classifier__C': 10}

Accuracy: 0.684

	precision	recall	f1-score	support
0	0.72	0.15	0.25	1393
1	0.68	0.97	0.80	2607
accuracy			0.68	4000
macro avg	0.70	0.56	0.52	4000
weighted avg	0.70	0.68	0.61	4000

Random Forest Before Feature Selection

Best Params: {'classifier__max_depth': 10, 'classifier__min_samples_split': 2, 'classifier__n

Accuracy: 0.7005

	precision	recall	f1-score	support
0	0.69	0.26	0.37	1393
1	0.70	0.94	0.80	2607
accuracy			0.70	4000
macro avg	0.70	0.60	0.59	4000
weighted avg	0.70	0.70	0.65	4000

```
X_train_processed = preprocessor.fit_transform(X_train)
```

```
X_test_processed = preprocessor.transform(X_test)
```

```
feature_names = preprocessor.get_feature_names_out()
```

```
selector_kbest = SelectKBest(score_func=mutual_info_classif, k=15)
```

```

selector_kbest = SelectKBest(score_func=mutual_info_classif, k=15)
X_train_kbest = selector_kbest.fit_transform(X_train_processed, y_train)
X_test_kbest = selector_kbest.transform(X_test_processed)
kbest_features = feature_names[selector_kbest.get_support()]

selector_rfe = RFE(LogisticRegression(max_iter=1000), n_features_to_select=15)
X_train_rfe = selector_rfe.fit_transform(X_train_processed, y_train)
X_test_rfe = selector_rfe.transform(X_test_processed)
rfe_features = feature_names[selector_rfe.get_support()]

print('Top features using SelectKBest:')
print(kbest_features)

print('\nTop features using RFE:')
print(rfe_features)

results_after_fs = []

fs_datasets = {
    'SelectKBest': (X_train_kbest, X_test_kbest),
    'RFE': (X_train_rfe, X_test_rfe)
}

for fs_name, (Xtr, Xts) in fs_datasets.items():
    print(f'\n==== Results using {fs_name} =====')

    lr_grid = GridSearchCV(
        LogisticRegression(max_iter=1000),
        {'C': [0.1, 1, 10]},
        cv=5,
        scoring='accuracy',
        n_jobs=-1
    )
    lr_grid.fit(Xtr, y_train)
    lr_pred = lr_grid.predict(Xts)

    results_after_fs.append({
        'Feature Selection': fs_name,
        'Model': 'Logistic Regression',
        'Best Params': lr_grid.best_params_,
        'Accuracy': accuracy_score(y_test, lr_pred)
    })

    print('\nLogistic Regression')
    print('Best Params:', lr_grid.best_params_)
    print('Accuracy:', accuracy_score(y_test, lr_pred))
    print(classification_report(y_test, lr_pred))

    rf_grid = GridSearchCV(
        RandomForestClassifier(random_state=42),
        {
            'n_estimators': [100, 200],
            'max_depth': [None, 10, 20],
            'min_samples_split': [2, 5]
        },
        cv=5,
        scoring='accuracy',
        n_jobs=-1
    )
    rf_grid.fit(Xtr, y_train)
    rf_pred = rf_grid.predict(Xts)

    results_after_fs.append({
        'Feature Selection': fs_name,
        'Model': 'Random Forest',

```

```
'Best Params': rf_grid.best_params_,
'Accuracy': accuracy_score(y_test, rf_pred)
})

print('\nRandom Forest')
print('Best Params:', rf_grid.best_params_)
print('Accuracy:', accuracy_score(y_test, rf_pred))
print(classification_report(y_test, rf_pred))

before_fs_df = pd.DataFrame(results_before_fs)
after_fs_df = pd.DataFrame(results_after_fs)

print('\nResults Before Feature Selection:')
print(before_fs_df)

print('\nResults After Feature Selection:')
print(after_fs_df)
```

0	0.69	0.25	0.37	1393
1	0.70	0.94	0.80	2607
accuracy			0.70	4000
macro avg	0.69	0.60	0.59	4000
weighted avg	0.70	0.70	0.65	4000

==== Results using RFE ====

Logistic Regression
Best Params: {'C': 10}
Accuracy: 0.6715

5 RFE Random Forest

	Best Params	Accuracy
0	{'C': 10}	0.68125
1	{'max_depth': 10, 'min_samples_split': 2, 'n_e...	0.70000
2	{'C': 10}	0.67150
3	{'max_depth': None, 'min_samples_split': 2, 'n...	0.69250