```
# --- Install required libraries ---
!pip install -q scikit-learn matplotlib pandas numpy joblib mlflow evidently fastapi uvicorn

# --- Import packages ---
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, roc_curve,
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.0/40.0 kB 1.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.9/8.9 MB 52.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 74.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 34.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.3/4.3 MB 56.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 147.8/147.8 kB 8.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 237.8/237.8 kB 14.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.9/114.9 kB 8.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.0/85.0 kB 6.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.8/76.8 kB 5.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 564.6/564.6 kB 29.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.4/57.4 kB 3.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 753.9/753.9 kB 34.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 11.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 517.7/517.7 kB 23.6 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 213.6/213.6 kB 13.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 63.7/63.7 kB 4.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 75.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 456.8/456.8 kB 21.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 70.2/70.2 kB 4.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 63.7 MB/s eta 0:00:00
```

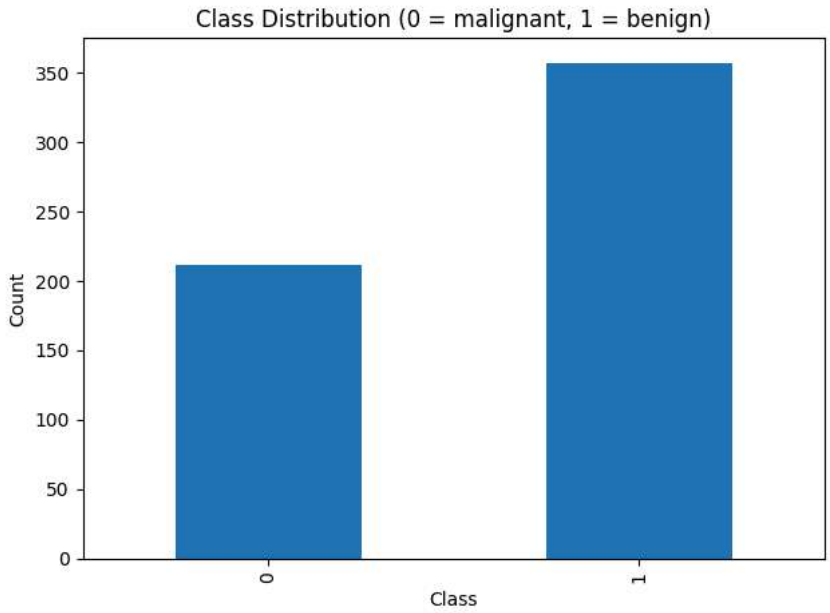
```
# Load dataset
data_bunch = load_breast_cancer(as_frame=True)
df = data_bunch.frame.copy()
df.rename(columns={"target": "label"}, inplace=True)

# Clean and split
df = df.drop_duplicates().reset_index(drop=True)
X = df.drop(columns=["label"])
y = df["label"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

df.head()
```

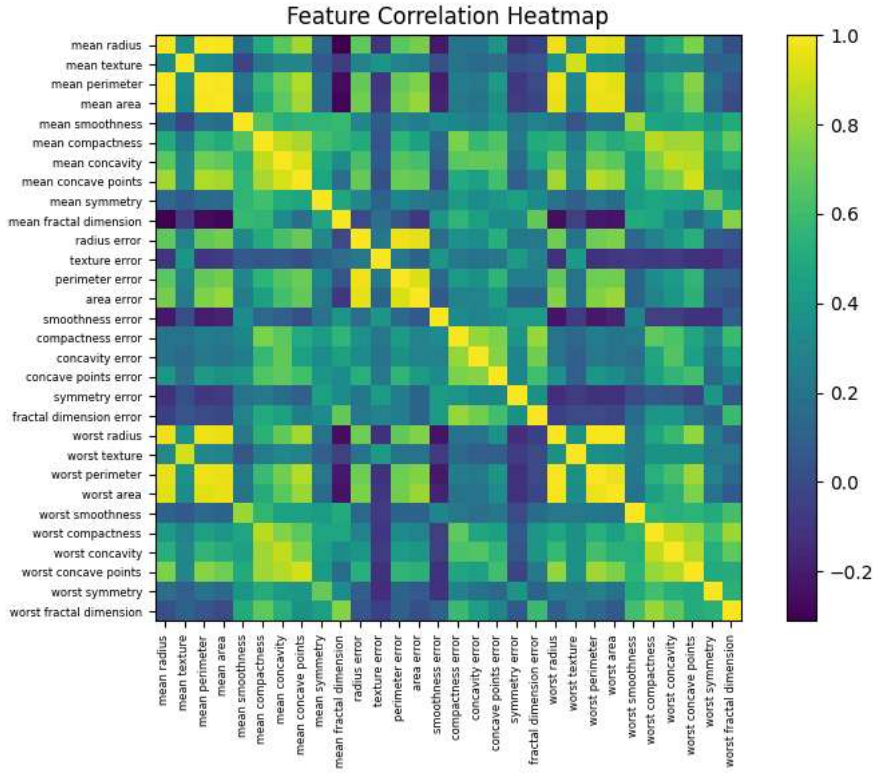
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20

5 rows × 31 columns

```
# Class distribution
plt.figure()
y.value_counts().sort_index().plot(kind="bar")
```

```
plt.title("Class Distribution (0 = malignant, 1 = benign)")
plt.xlabel("Class")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

# Correlation heatmap (matplotlib only)
corr = X.corr(numeric_only=True).values
labels = list(X.columns)
plt.figure(figsize=(8, 6))
plt.imshow(corr, interpolation="nearest")
plt.title("Feature Correlation Heatmap")
plt.colorbar()
plt.xticks(range(len(labels)), labels, rotation=90, fontsize=6)
plt.yticks(range(len(labels)), labels, fontsize=6)
plt.tight_layout()
plt.show()
```



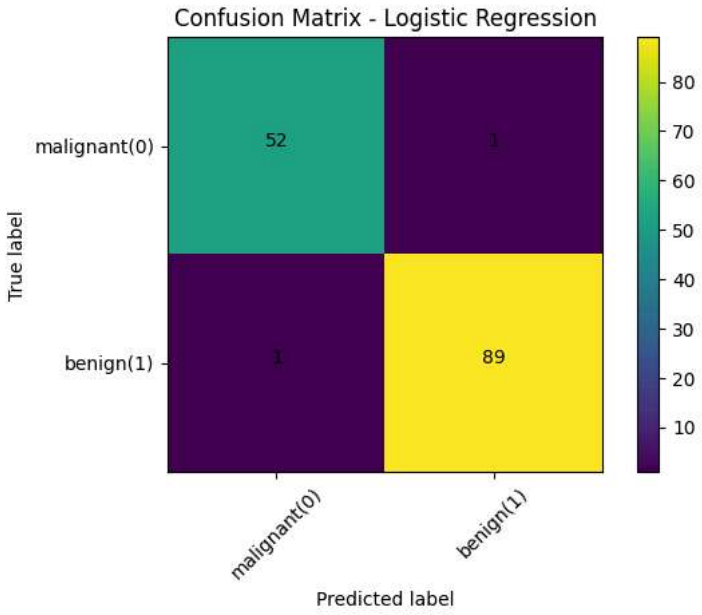


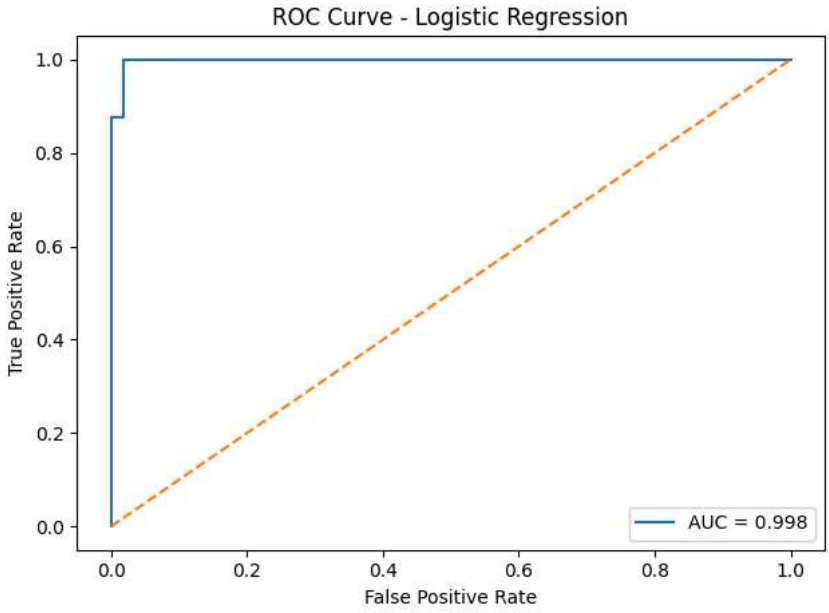
```
models = {
    "Logistic Regression": Pipeline([
        ("scaler", StandardScaler()),
        ("clf", LogisticRegression(max_iter=1000, solver="lbfgs"))
    ]),
    "SVM (RBF)": Pipeline([
        ("scaler", StandardScaler()),
        ("clf", SVC(kernel="rbf", probability=True))
    ]),
    "Random Forest": RandomForestClassifier(n_estimators=300, random_state=42)
}

results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, "predict_proba") else y_pred.astype(float)

    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    rocauc = roc_auc_score(y_test, y_proba)

    results.append({"model": name, "accuracy": acc, "precision": prec, "recall": rec, "f1": f1, "roc_auc": rocauc})

pd.DataFrame(results).sort_values(by=["roc_auc", "f1", "accuracy"], ascending=False)
```

	model	accuracy	precision	recall	f1	roc_auc
0	Logistic Regression	0.986014	0.988889	0.988889	0.988889	0.997694
1	SVM (RBF)	0.979021	0.988764	0.977778	0.983240	0.996855
2	Random Forest	0.958042	0.956522	0.977778	0.967033	0.994864

```
import itertools

def plot_cm(cm, classes, title='Confusion matrix'):
    plt.figure()
    plt.imshow(cm, interpolation='nearest')
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'd'), horizontalalignment="center")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

# Select best model by ROC-AUC
best = sorted(results, key=lambda d: (d["roc_auc"], d["f1"], d["accuracy"]), reverse=True)[0]
best_name = best["model"]
best_model = models[best_name].fit(X_train, y_train)
y_pred = best_model.predict(X_test)
y_proba = best_model.predict_proba(X_test)[:, 1]

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plot_cm(cm, ["malignant(0)", "benign(1)"], title=f"Confusion Matrix - {best_name}")
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
plt.figure()
plt.plot(fpr, tpr, label=f"AUC = {auc(fpr, tpr):.3f}")
plt.plot([0,1],[0,1], linestyle="--")
plt.title(f"ROC Curve - {best_name}")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend(loc="lower right")
plt.tight_layout()
```

```
plt.show()
```






```
print(f"Best Model: {best_name}")
print("""
Interpretation:
- Best model chosen using ROC-AUC and F1 for balance between sensitivity and precision.
- Confusion matrix reveals type of misclassifications.
- Correlation heatmap suggests potential redundancy among features; tree models handle that well.

Deployment Plan:
- Export pipeline: joblib.dump(best_model, "model.pkl")
- Serve with FastAPI endpoint `/predict` inside Docker container
- Monitor input drift and prediction quality with Evidently + MLflow
- Retrain periodically when performance decays
""")
```

```
Best Model: Logistic Regression

Interpretation:
- Best model chosen using ROC-AUC and F1 for balance between sensitivity and precision.
- Confusion matrix reveals type of misclassifications.
- Correlation heatmap suggests potential redundancy among features; tree models handle that well.

Deployment Plan:
- Export pipeline: joblib.dump(best_model, "model.pkl")
```

```
- Serve with FastAPI endpoint `/predict` inside Docker container
- Monitor input drift and prediction quality with Evidently + MLflow
- Retrain periodically when performance decays
```