```
!pip install imblearn joblib seaborn

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import logging
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchC
from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeature
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, PolynomialFeature
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import (classification_report, accuracy_score, f1_score,
                             roc_auc_score, confusion_matrix, roc_curve,
                             precision_recall_curve, precision_score, recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, HistGradientBoostingClassifie
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline
import joblib

```

```
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl.metadata (355 bytes)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (1.5.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.12/dist-packages (from imblearn) (0.14.0)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.12/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->se
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seabor
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->s
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->sea
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4->se
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib!=3.6.1,>=3.4
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2
Requirement already satisfied: scipy<2,>=1.11.4 in /usr/local/lib/python3.12/dist-packages (from imbalanced-learn->imblearn)
Requirement already satisfied: scikit-learn<2,>=1.4.2 in /usr/local/lib/python3.12/dist-packages (from imbalanced-learn->imb
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from imbalanced-learn->im
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib!=3
Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

File Import from PC

```
from google.colab import files

uploaded = files.upload()

# Get uploaded filename
file_name = list(uploaded.keys())[0]

# Read Excel file
df = pd.read_excel(file_name)
df.head()
```

11

Choose files osteoarthritis_dataset.xlsx
osteoarthritis_dataset.xlsx(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 7092587 bytes, last modified: 25/10/2025 - 100% done
Saving osteoarthritis_dataset.xlsx to osteoarthritis_dataset.xlsx

	patient_id	age	sex	bmi	knee_side	cartilage_medial_mm	cartilage_lateral_mm	joint_space_medial_mm	joint_space_l
0	1	78	Female	32.35	Right	2.09	2.52	2.88	
1	2	68	Male	28.86	Right	3.06	3.74	2.25	
2	3	54	Female	27.47	Right	3.68	2.85	2.16	
3	4	82	Female	22.56	Left	2.73	4.84	2.61	
4	5	47	Female	31.24	Right	3.46	3.47	1.88	

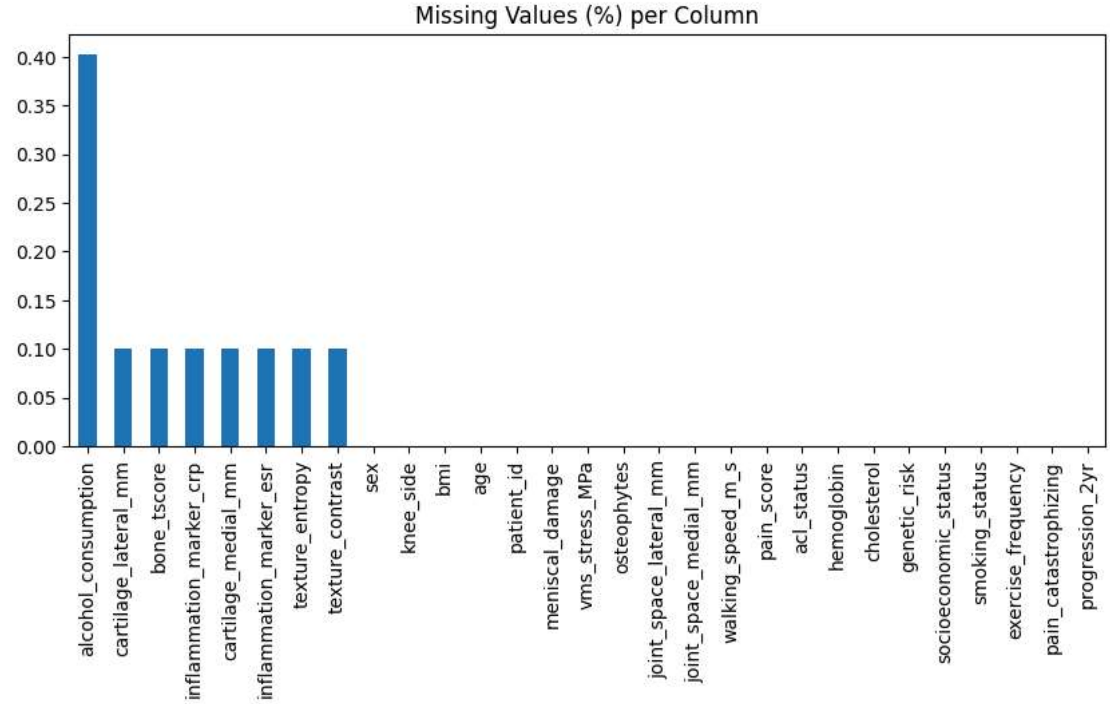
5 rows × 29 columns

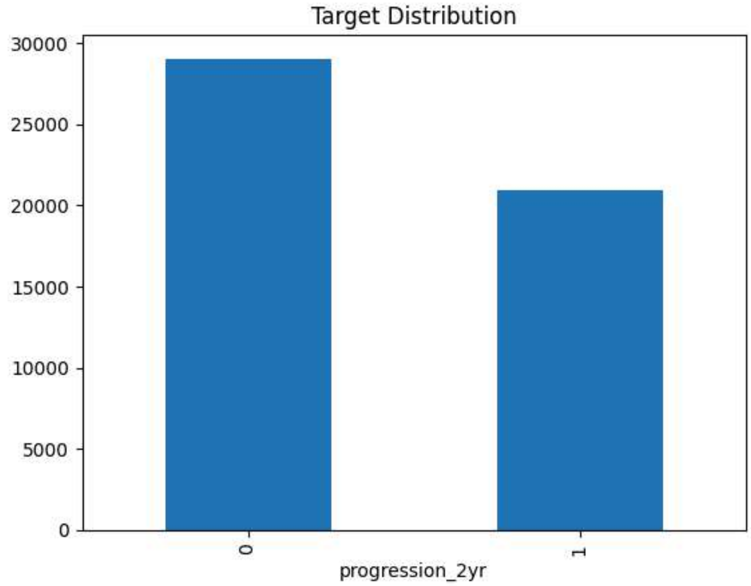
Basic Checks

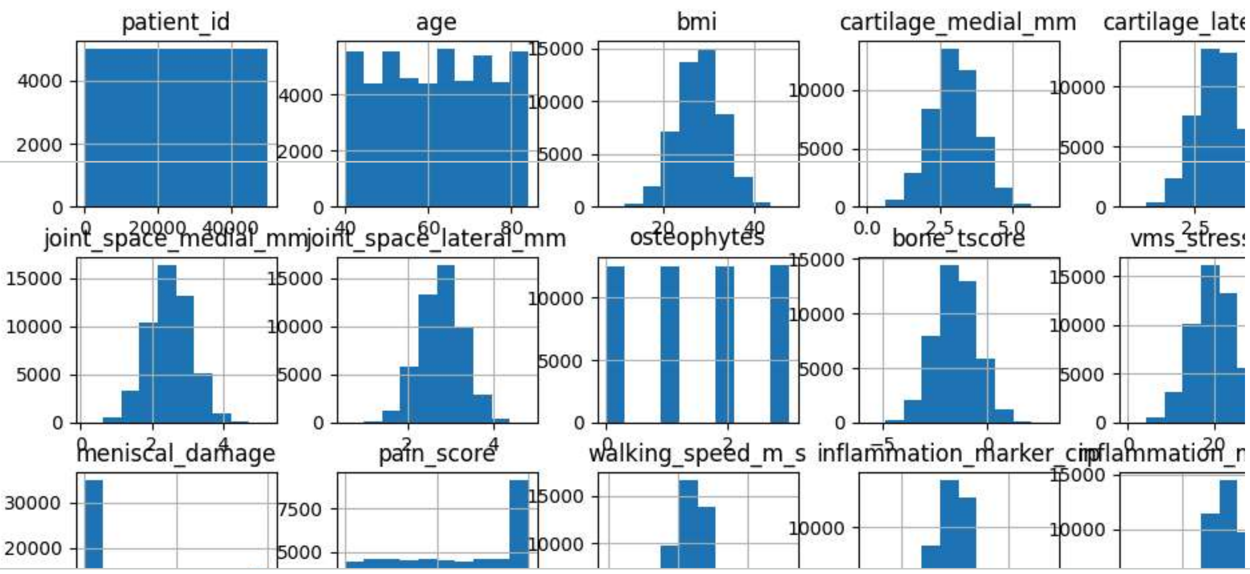
```
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

assert df.shape[0] > 0 and df.shape[1] > 0, "Empty dataset"
logger.info(f"Dataset shape: {df.shape}")

print(df.shape)
print(df.info())
print(df.describe())

# Missing values
df.isnull().sum().sort_values(ascending=False).head()

plt.figure(figsize=(10,4))
df.isnull().mean().sort_values(ascending=False).plot(kind="bar")
plt.title("Missing Values (%) per Column")
plt.show()

# Class balance
df['progression_2yr'].value_counts().plot(kind="bar")
plt.title("Target Distribution")
plt.show()

# Numeric distributions
df.select_dtypes(include='number').hist(figsize=(12,10))
plt.show()

# Correlation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(numeric_only=True), annot=False, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

```

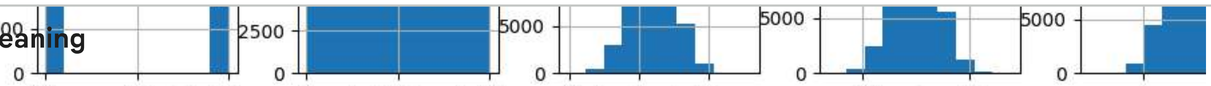
```
(50000, 29)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 29 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   patient_id               50000 non-null  int64  
 1   age                      50000 non-null  int64  
 2   sex                      50000 non-null  object 
 3   bmi                      50000 non-null  float64
 4   knee_side                50000 non-null  object 
 5   cartilage_medial_mm      45000 non-null  float64
 6   cartilage_lateral_mm     45000 non-null  float64
 7   joint_space_medial_mm    50000 non-null  float64
 8   joint_space_lateral_mm   50000 non-null  float64
 9   osteophytes              50000 non-null  int64  
 10  bone_tscore              45000 non-null  float64
 11  vms_stress_MPa           50000 non-null  float64
 12  meniscal_damage          50000 non-null  int64  
 13  acl_status               50000 non-null  object 
 14  pain_score               50000 non-null  int64  
 15  walking_speed_m_s        50000 non-null  float64
 16  inflammation_marker_crp  45000 non-null  float64
 17  inflammation_marker_esr  45000 non-null  float64
 18  genetic_risk             50000 non-null  int64  
 19  cholesterol              50000 non-null  float64
 20  hemoglobin               50000 non-null  float64
 21  socioeconomic_status     50000 non-null  object 
 22  exercise_frequency       50000 non-null  int64  
 23  smoking_status           50000 non-null  object 
 24  alcohol_consumption      29852 non-null  object 
 25  texture_entropy          45000 non-null  float64
 26  texture_contrast         45000 non-null  float64
 27  pain_catastrophizing     50000 non-null  int64  
 28  progression_2yr          50000 non-null  int64  
dtypes: float64(14), int64(9), object(6)
memory usage: 11.1+ MB
None
         patient_id          age           bmi  cartilage_medial_mm  \
count  50000.000000  50000.00000  50000.000000         45000.000000   
mean   25000.500000     61.98650     27.987705             3.006427   
std    14433.901067     12.95123      4.985826             0.800577   
min        1.000000     40.00000      7.500000             0.010000   
25%    12500.750000     51.00000     24.640000             2.470000   
50%    25000.500000     62.00000     27.980000             3.000000   
75%    37500.250000     73.00000     31.350000             3.540000   
max    50000.000000     84.00000     47.500000             6.260000   

       cartilage_lateral_mm  joint_space_medial_mm  joint_space_lateral_mm  \
count          45000.000000           50000.000000            50000.000000   
mean               3.199000               2.500796                2.799205   
std                0.699206               0.598143                0.497948   
min                0.440000               0.110000                0.560000   
25%                2.720000               2.100000                2.460000   
50%                3.200000               2.500000                2.800000   
75%                3.670000               2.910000                3.140000   
max                6.050000               5.250000                4.820000   

        osteophytes   bone_tscore  vms_stress_MPa  ...  \
count  50000.000000  45000.000000    50000.000000  ...   
mean       1.504840     -1.508637       20.019953  ...   
std        1.118553      1.005187        5.015887  ...   
min        0.000000     -5.750000        0.040000  ...   
25%        1.000000     -2.190000       16.640000  ...   
50%        2.000000     -1.510000       20.000000  ...   
75%        3.000000     -0.830000       23.380000  ...   
max        3.000000      2.980000       42.500000  ...   

       inflammation_marker_crp  inflammation_marker_esr  genetic_risk  \
count             45000.000000             45000.000000  50000.000000   
mean                  4.991103                20.032799      0.198020   
std                   1.999562                10.025325      0.398511   
min                  -3.190000               -25.270000      0.000000   
25%                   3.630000                13.217500      0.000000   
50%                   5.000000                20.010000      0.000000   
75%                   6.330000                26.840000      0.000000   
max                  13.850000                58.880000      1.000000   

        cholesterol    hemoglobin  exercise_frequency  texture_entropy  \
count  50000.000000  50000.000000        50000.000000     45000.000000   
mean     199.732311     13.499771            2.498020         0.799940   
std       40.105047      1.500007            1.699369         0.198822   
min       14.870000      7.150000            0.000000        -0.009000   
25%      172.740000     12.490000            1.000000         0.666000   
50%      199.680000     13.490000            2.000000         0.801000   
75%      226.740000     14.500000            4.000000         0.935000   
max      372.970000     19.970000            5.000000         1.588000   

       texture_contrast  pain_catastrophizing  progression_2yr  
count      45000.000000          50000.000000     50000.000000  
```

```
mean      1.499863     9.965480     0.419140
std       0.403010     6.069371     0.493423
min      -0.224000     0.000000     0.000000
25%       1.228000     5.000000     0.000000
50%       1.497000    10.000000     0.000000
75%       1.774000    15.000000     1.000000
max       3.332000    20.000000     1.000000

[8 rows x 23 columns]
```



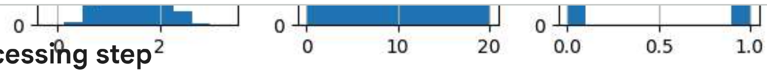






Data Cleaning

```
# Remove duplicates
df = df.drop_duplicates()

# Select target
target = 'progression_2yr'
X = df.drop(columns=[target, 'patient_id'])
y = df[target].astype(int)

# Numeric & categorical columns


```



Preprocessing step

Correlation Heatmap

```
target = 'progression_2yr'
X = df.drop(columns=[target, 'patient_id'])
y = df[target].astype(int)


X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,stratify=y,random
logger.info(f'Training shape : {X_train.shape} Testing shape : {X_test.shape}')


numeric_cols = X.select_dtypes(include='number').columns.tolist()
cat_cols = X.select_dtypes(include=['object','category']).columns.tolist()

num_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("scaler", StandardScaler())
])

cat_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("onehot", OneHotEncoder(handle_unknown="ignore"))
])

preprocessor = ColumnTransformer([
    ("num", num_pipeline, numeric_cols),
    ("cat", cat_pipeline, cat_cols)
])

```

Model loop

```
models = {
    "LogisticRegression": {
        "model": LogisticRegression(max_iter=2000, solver="saga"),
        "params": {"model__C": [10]}
    },
    "RandomForest": {
        "model": RandomForestClassifier(),
        "params": {"model__n_estimators": [5,10], "model__max_depth": [1,10]}
    },
    "HistBoost": {
        "model": HistGradientBoostingClassifier(),
        "params": {"model__max_iter": [100], "model__max_depth": [10]}
```

```
    }
}

results = {}
skf = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)

for name, mp in models.items():
    pipe = ImbPipeline([
        ("preproc", preprocessor),
        ("smote", SMOTE()),
        ("model", mp["model"])
    ])

    grid = GridSearchCV(pipe, mp["params"], cv=skf, scoring="f1", n_jobs=-1)
    grid.fit(X, y)

    best_model = grid.best_estimator_
    y_pred = best_model.predict(X)
    y_proba = best_model.predict_proba(X)[:,1]

    results[name] = {
        "model": best_model,
        "f1": f1_score(y, y_pred),
        "roc": roc_auc_score(y, y_proba)
    }

    print(f"{name} = {results[name]}")

```

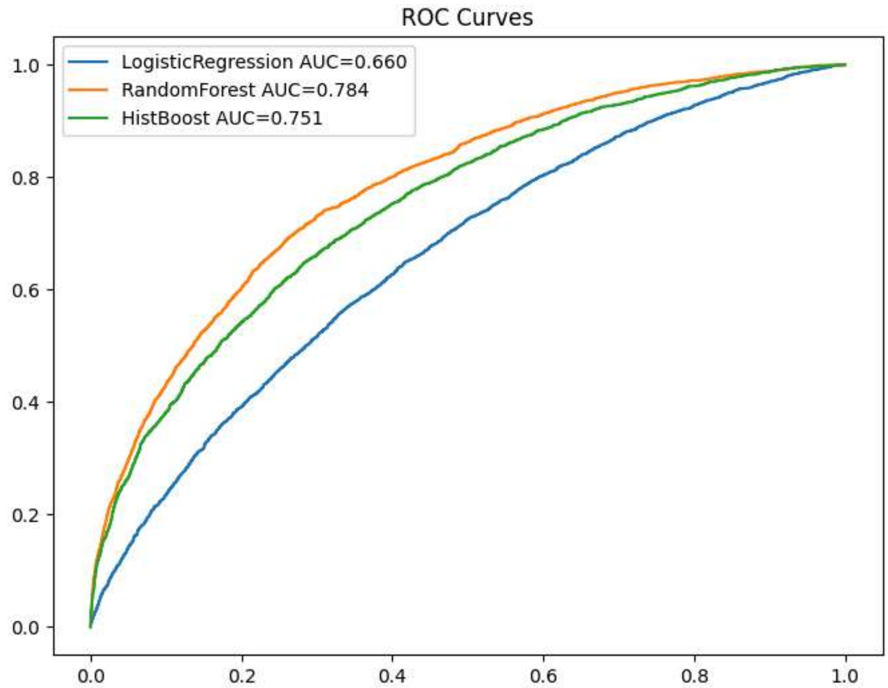
```
                ('smote', SMOTE()),
                ('model',
                 LogisticRegression(C=10, max_iter=2000, solver='saga'))]), 'f1': 0.5788680399309727, 'roc': np.float64(0.
RandomForest = {'model': Pipeline(steps=[('preproc',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('poly',
                                                                   PolynomialFeatures(include_bias=False)),
                                                                  ('scaler',
                                                                   StandardScaler())]),
                                                  ['age', 'bmi',
                                                   'cartilage_medial_mm',
                                                   'cartilage_lateral_mm',
                                                   'joint_space_medial_mm',
                                                   'joint_space_lateral_mm',
                                                   'osteophytes', 'bone_tscore',
                                                   'vms_stres...
                                                   'texture_contrast',
                                                   'pain_catastrophizing']),
                                                 ('cat',
                                                  Pipeline(steps=[('imputer',
```

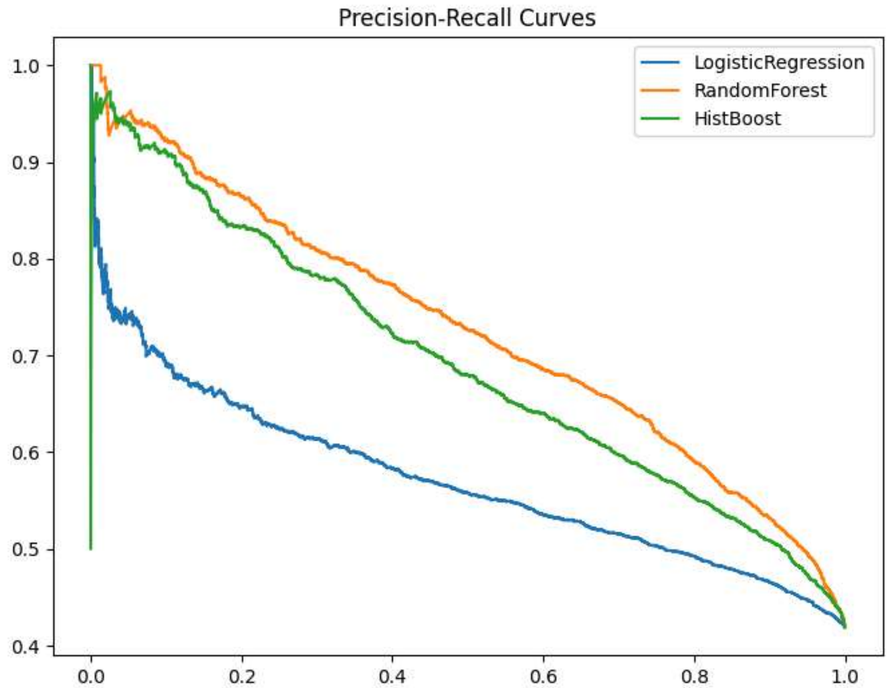
```
                                                   'texture_contrast'
```

```
                                  'texture_contrast',
                                  'pain_catastrophizing']),
                                ('cat',
                                 Pipeline(steps=[('imputer',
                                                  SimpleImputer(strategy='most_frequent')),
                                                 ('onehot',
                                                  OneHotEncoder(handle_unknown='ignore'))]),
                                 ['sex', 'knee_side',
```

ROC & PR Curves

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
for name in results:
    model = results[name]["model"]
    probs = model.predict_proba(X_test)[:,1]
    fpr, tpr,_ = roc_curve(y_test, probs)
    auc_val = roc_auc_score(y_test, probs)
    plt.plot(fpr, tpr, label=f"{name} AUC={auc_val:.3f}")
plt.legend()
plt.title("ROC Curves")
plt.show()

plt.figure(figsize=(8,6))
for name in results:
    probs = results[name]["model"].predict_proba(X_test)[:,1]
    prec, rec,_ = precision_recall_curve(y_test, probs)
    plt.plot(rec, prec, label=name)
plt.legend()
plt.title("Precision-Recall Curves")
plt.show()

```





Model Comparison Table

```
comparison = pd.DataFrame(results).T
comparison = comparison[['acc','f1','roc','prec','rec','cv_f1_mean']]
print("MODEL COMPARISON:\n")
print(comparison)

comparison.to_csv("model_comparison.csv")

```

Double-click (or enter) to edit

Feature Importance (RF Only)

```
rf = results["RandomForest"]["model"]
feat_names = rf.named_steps["preproc"].get_feature_names_out()
importances = rf.named_steps["model"].feature_importances_
```