```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, fbeta_score, roc_auc_score, confusion_matrix, roc_curve
)

import matplotlib.pyplot as plt
import seaborn as sns

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.neural_network import MLPClassifier

train = pd.read_csv('train_dataset.csv')
df=train.copy()
```

```
print(train.shape)
print(train.head())
print(train.info())
```

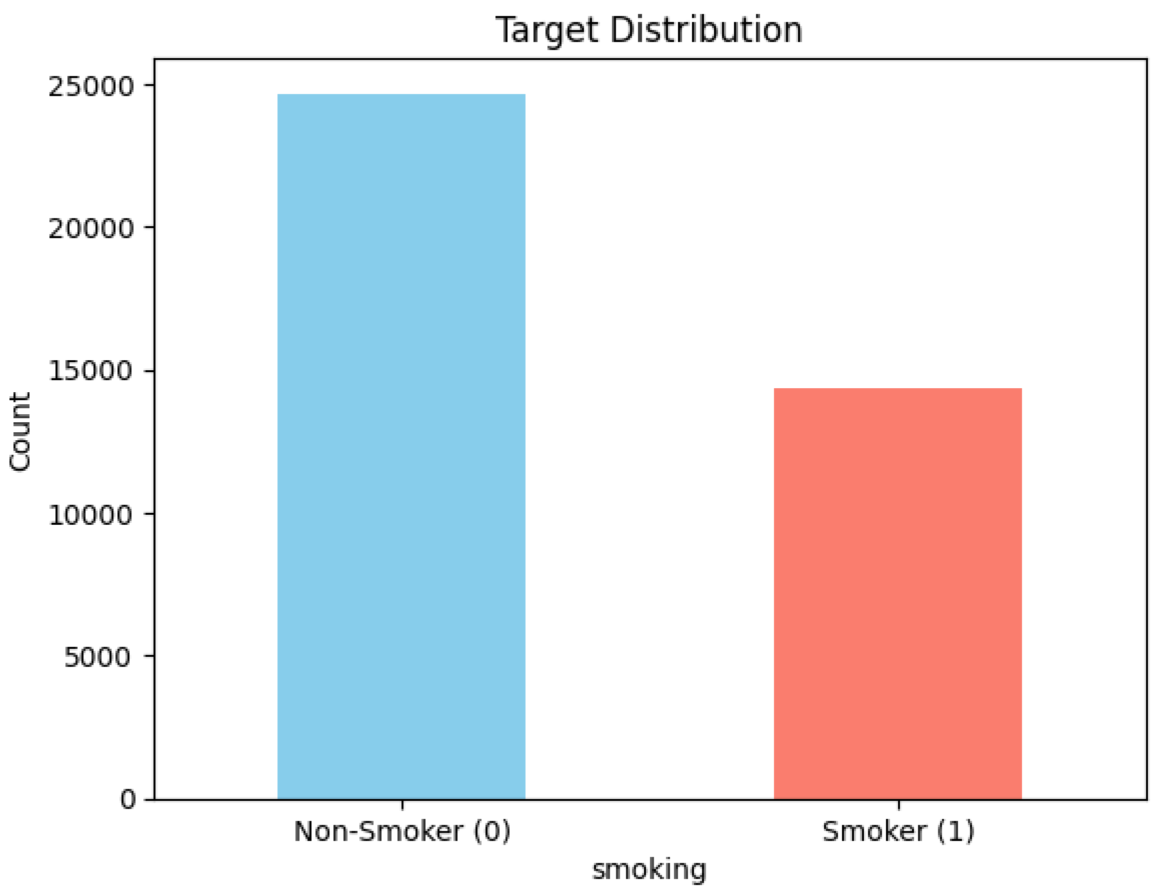
```
2   45         155          65       86.0             0.9              0.9
3   45         165          80       94.0             0.8              0.7
4   20         165          60       81.0             1.5              0.1

   hearing(left)  hearing(right)  systolic  relaxation  ...  HDL  LDL  \
0              1               1       118          78  ...   70  142
1              1               1       119          79  ...   71  114
2              1               1       110          80  ...   57  112
3              1               1       158          88  ...   46   91
4              1               1       109          64  ...   47   92

   hemoglobin  Urine protein  serum creatinine   AST   ALT  Gtp  \
0        19.8              1               1.0    61   115  125
1        15.9              1               1.1    19    25   30
2        13.7              3               0.6  1090  1400  276
3        16.9              1               0.9    32    36   36
4        14.9              1               1.2    26    28   15
```

```
2             0        0
3             0        0
4             0        0

[5 rows x 23 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38984 entries, 0 to 38983
Data columns (total 23 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   age                  38984 non-null  int64  
 1   height(cm)           38984 non-null  int64  
 2   weight(kg)           38984 non-null  int64  
 3   waist(cm)            38984 non-null  float64
 4   eyesight(left)       38984 non-null  float64
 5   eyesight(right)      38984 non-null  float64
 6   hearing(left)        38984 non-null  int64  
 7   hearing(right)       38984 non-null  int64  
 8   systolic             38984 non-null  int64  
 9   relaxation           38984 non-null  int64  
 10  fasting blood sugar  38984 non-null  int64  
 11  Cholesterol          38984 non-null  int64  
 12  triglyceride         38984 non-null  int64  
 13  HDL                  38984 non-null  int64  
 14  LDL                  38984 non-null  int64  
 15  hemoglobin           38984 non-null  float64
 16  Urine protein        38984 non-null  int64  
 17  serum creatinine     38984 non-null  float64
 18  AST                  38984 non-null  int64  
 19  ALT                  38984 non-null  int64  
 20  Gtp                  38984 non-null  int64  
 21  dental caries        38984 non-null  int64  
 22  smoking              38984 non-null  int64  
dtypes: float64(5), int64(18)
memory usage: 6.8 MB
None
```

```
train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	38984.0	44.127591	12.063564	20.0	40.0	40.0	55.0	85.0
height(cm)	38984.0	164.689488	9.187507	130.0	160.0	165.0	170.0	190.0
weight(kg)	38984.0	65.938718	12.896581	30.0	55.0	65.0	75.0	135.0
waist(cm)	38984.0	82.062115	9.326798	51.0	76.0	82.0	88.0	129.0
eyesight(left)	38984.0	1.014955	0.498527	0.1	0.8	1.0	1.2	9.9
eyesight(right)	38984.0	1.008768	0.493813	0.1	0.8	1.0	1.2	9.9
hearing(left)	38984.0	1.025369	0.157246	1.0	1.0	1.0	1.0	2.0
hearing(right)	38984.0	1.026190	0.159703	1.0	1.0	1.0	1.0	2.0
systolic	38984.0	121.475631	13.643521	71.0	112.0	120.0	130.0	233.0
relaxation	38984.0	75.994408	9.658734	40.0	70.0	76.0	82.0	146.0
fasting blood sugar	38984.0	99.342269	20.642741	46.0	89.0	96.0	104.0	423.0
Cholesterol	38984.0	196.883491	36.353945	55.0	172.0	195.0	219.0	445.0
triglyceride	38984.0	126.749461	71.803143	8.0	74.0	108.0	160.0	999.0
HDL	38984.0	57.293146	14.617822	4.0	47.0	55.0	66.0	359.0
LDL	38984.0	115.081495	42.883163	1.0	91.0	113.0	136.0	1860.0
hemoglobin	38984.0	14.624264	1.566528	4.9	13.6	14.8	15.8	21.1
Urine protein	38984.0	1.086523	0.402107	1.0	1.0	1.0	1.0	6.0
serum creatinine	38984.0	0.886030	0.220621	0.1	0.8	0.9	1.0	11.6
AST	38984.0	26.198235	19.175595	6.0	19.0	23.0	29.0	1090.0
ALT	38984.0	27.145188	31.309945	1.0	15.0	21.0	31.0	2914.0
Gtp	38984.0	39.905038	49.693843	2.0	17.0	26.0	44.0	999.0
dental caries	38984.0	0.214421	0.410426	0.0	0.0	0.0	0.0	1.0

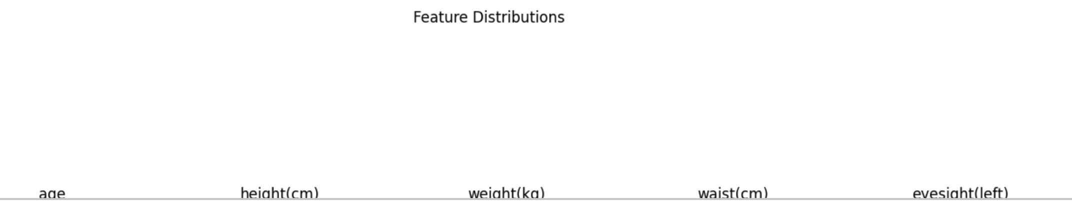
```
plt.figure()
train['smoking'].value_counts().plot(kind='bar', color=['skyblue', 'salmon']
plt.xticks([0, 1], ['Non-Smoker (0)', 'Smoker (1)'], rotation=0)
plt.ylabel("Count")
plt.title("Target Distribution")
plt.show()
```



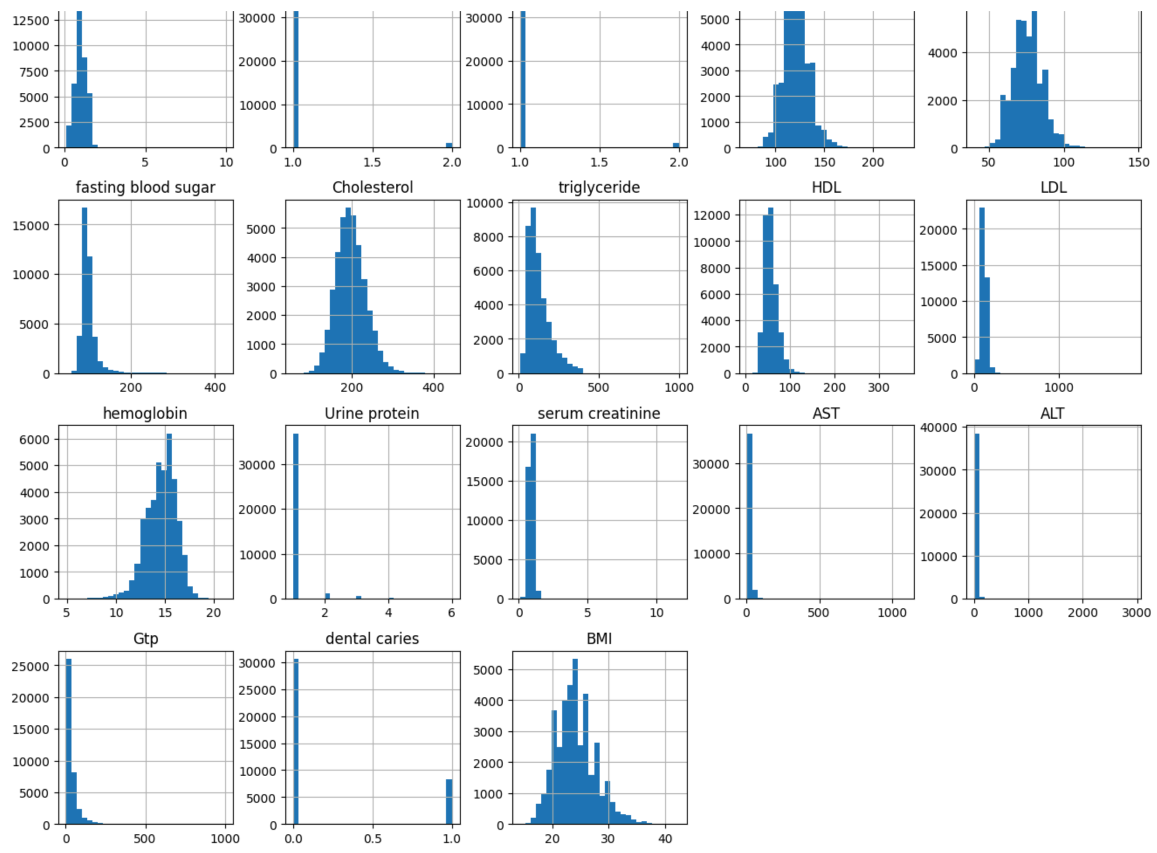
```
train['BMI'] = train['weight(kg)'] / (train['height(cm)'] / 100) ** 2
train[['weight(kg)', 'height(cm)', 'BMI']].head()
```

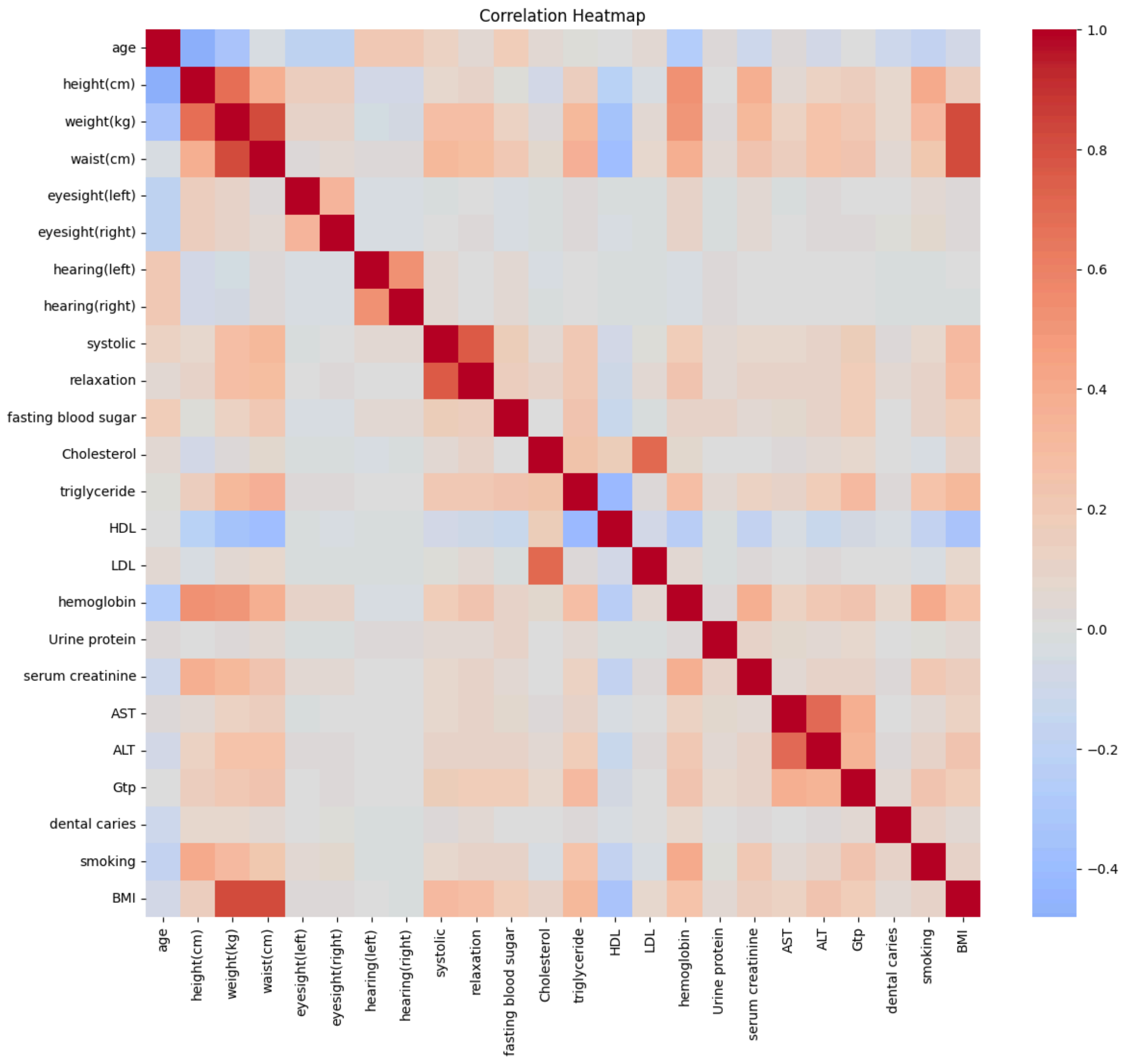
	weight(kg)	height(cm)	BMI
0	85	170	29.411765
1	110	175	35.918367
2	65	155	27.055151
3	80	165	29.384757
4	60	165	22.038567

```
num_cols = train.drop(columns=[ 'smoking']).select_dtypes(include=[np.number

train[num_cols].hist(bins=30, figsize=(16,16))
plt.suptitle("Feature Distributions", y=1.02)
plt.show()
```



```
plt.figure(figsize=(14, 12))
corr = train.corr()
sns.heatmap(corr, annot=False, cmap='coolwarm', center=0)
plt.title("Correlation Heatmap")
plt.show()
```





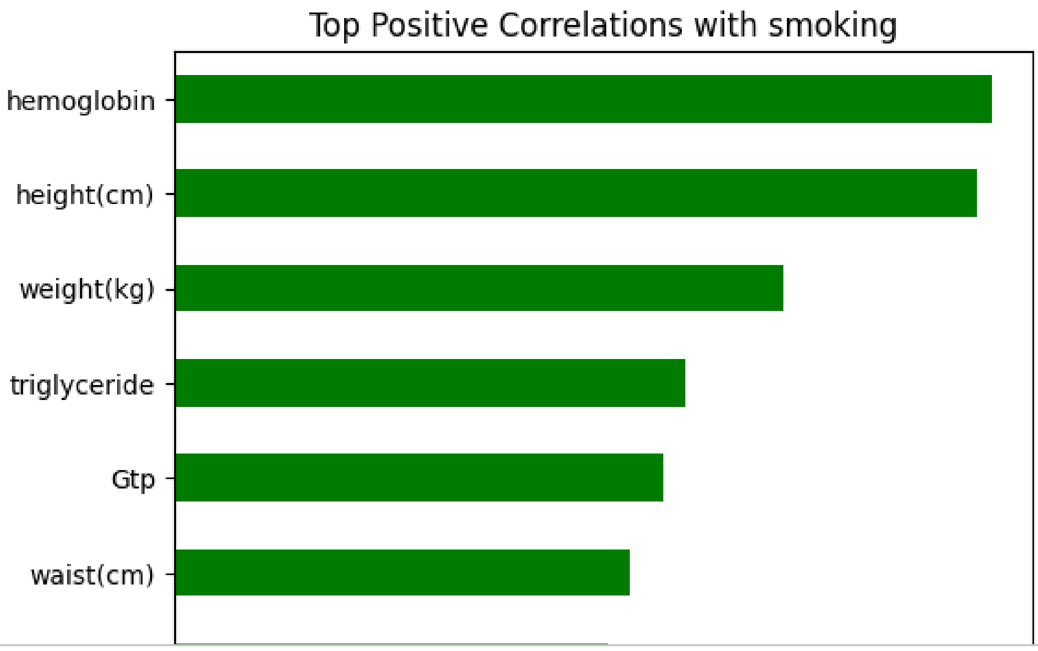
```
corr_target = corr['smoking'].sort_values(ascending=False)
corr_target
```

	smoking
smoking	1.000000
hemoglobin	0.401206
height(cm)	0.394314
weight(kg)	0.299347
triglyceride	0.251057
Gtp	0.240274
waist(cm)	0.223359
serum creatinine	0.212473
dental caries	0.107601
BMI	0.105488
relaxation	0.103663
fasting blood sugar	0.099908
ALT	0.098615
systolic	0.070176
eyesight(right)	0.064587
AST	0.062834
eyesight(left)	0.059409
Urine protein	0.013653
hearing(right)	-0.018990
hearing(left)	-0.022077
Cholesterol	-0.027493
LDL	-0.041627
age	-0.166268
HDL	-0.179509

dtype: float64

```
plt.figure(figsize=(6,8))
corr_target.drop('smoking').head(12).plot(kind='barh', color='green')
plt.title("Top Positive Correlations with smoking")
plt.gca().invert_yaxis()
plt.show()

plt.figure(figsize=(6,8))
corr_target.drop('smoking').tail(12).plot(kind='barh', color='red')
```

```
plt.title("Top Negative Correlations with smoking")
plt.gca().invert_yaxis()
plt.show()
```

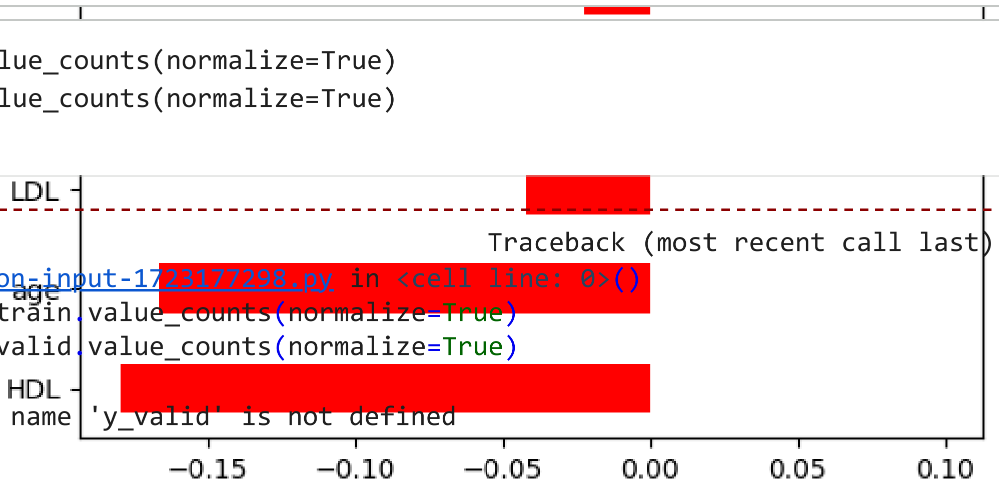


```
df["BMI"] = df["weight(kg)"] / (df["height(cm)"] / 100)**2

from sklearn.model_selection import train_test_split

X = df.drop(columns=["smoking", "id"], errors="ignore")
y = df["smoking"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.20,
    random_state=42,
    stratify=y
)

print("X_train:", X_train.shape)
print("X_test :", X_test.shape)
```

```
X_train: (31187, 23)
X_test : (7797, 23)
```

```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1]

    return {
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
        "Recall": recall_score(y_test, y_pred),
        "F1": f1_score(y_test, y_pred),
        "F2": fbeta_score(y_test, y_pred, beta=2),
        "ROC_AUC": roc_auc_score(y_test, y_prob),
        "CM": confusion_matrix(y_test, y_pred)
    }
```

```
y_train.value_counts(normalize=True)
y_valid.value_counts(normalize=True)
```



```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-1723177298.py in <cell line: 0>()
      1 y_train.value_counts(normalize=True)
----> 2 y_valid.value_counts(normalize=True)

NameError: name 'y_valid' is not defined
```

Next steps: Explain error

```
linear_svm = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", SVC(kernel="linear", probability=True))
])

linear_svm.fit(X_train, y_train)
lin_results = evaluate_model(linear_svm, X_test, y_test)
```

```
poly_svm = Pipeline([
    ("scaler", StandardScaler()),
    ("clf", SVC(kernel="poly", degree=2, C=5, gamma="scale", probability=Tru
])

poly_svm.fit(X_train, y_train)
poly_results = evaluate_model(poly_svm, X_test, y_test)
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('clf', SVC(kernel='poly', probability=True))
])

param_grid = {
    'clf__degree': [2, 3],
    'clf__C': [1, 5, 10, 15],
    'clf__gamma': [0.01, 0.05, 'scale']
}

grid = GridSearchCV(
    pipe,
    param_grid,
    cv=5,
    scoring='roc_auc',
    verbose=2
)

grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
print("Best AUC Score:", grid.best_score_)
```

Logistic Regression

```
    log_reg = Pipeline([
        ("scaler", StandardScaler()),
        ("clf", LogisticRegression(max_iter=500))
    ])

    log_reg.fit(X_train, y_train)
    log_results = evaluate_model(log_reg, X_test, y_test)
```

```
Start coding or generate with AI.
```

Neural Networks

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

    nn = Sequential([
        Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
        Dropout(0.2),
        Dense(32, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    nn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']

    es = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)

    nn.fit(
        X_train_scaled, y_train,
        validation_split=0.1,
        epochs=50,
        batch_size=32,
        callbacks=[es],
        verbose=0
    )
```