META in LR - Full model

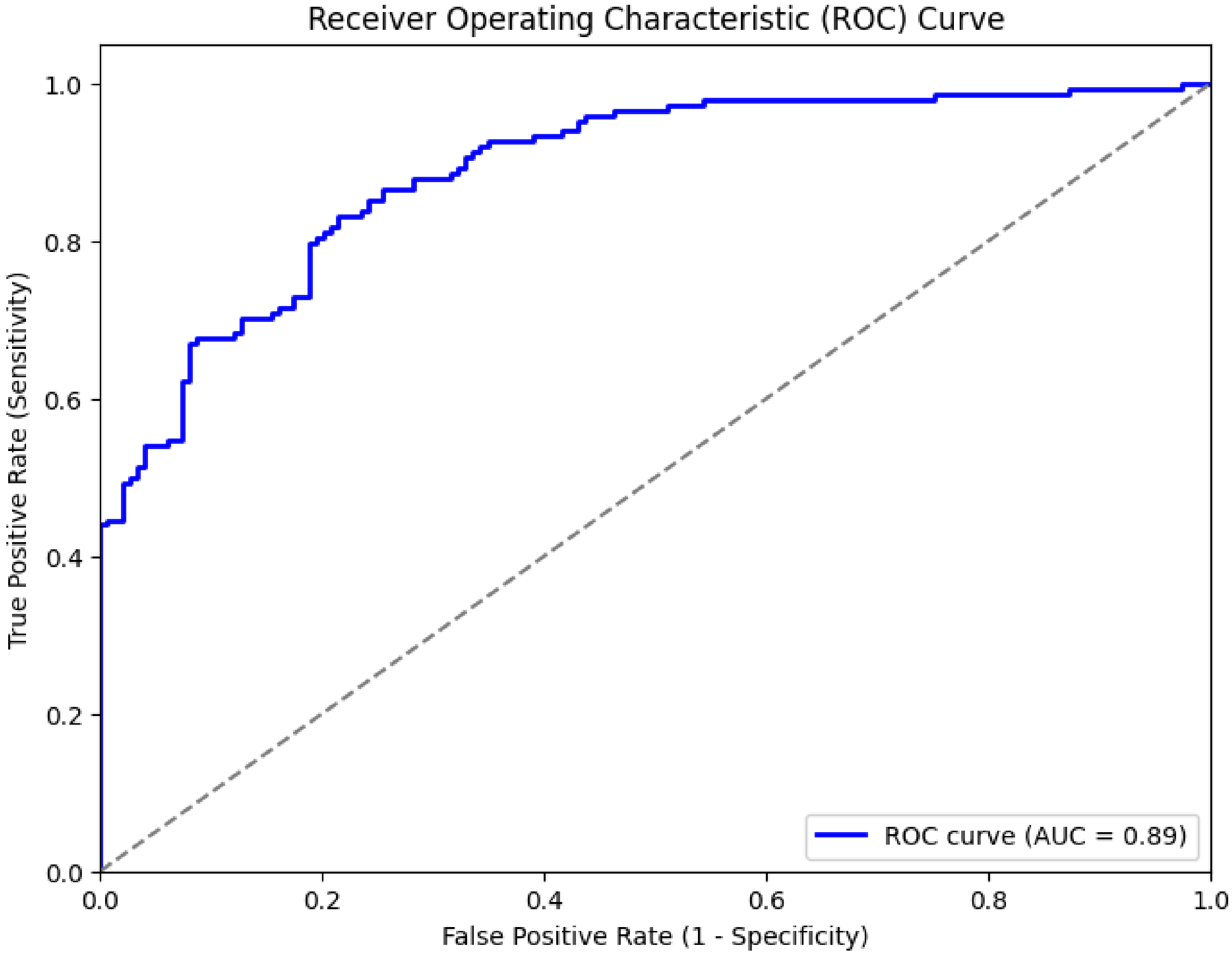
smote

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek


file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
```

```
plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.8193
  Fold 2: Accuracy = 0.8193
  Fold 3: Accuracy = 0.8059
  Fold 4: Accuracy = 0.8312
  Fold 5: Accuracy = 0.8059

Mean CV Accuracy: 0.8163

=== Final Test Set Evaluation ===
Accuracy: 0.8047
Precision: 0.7885
Recall: 0.8311
F1 Score: 0.8092
```



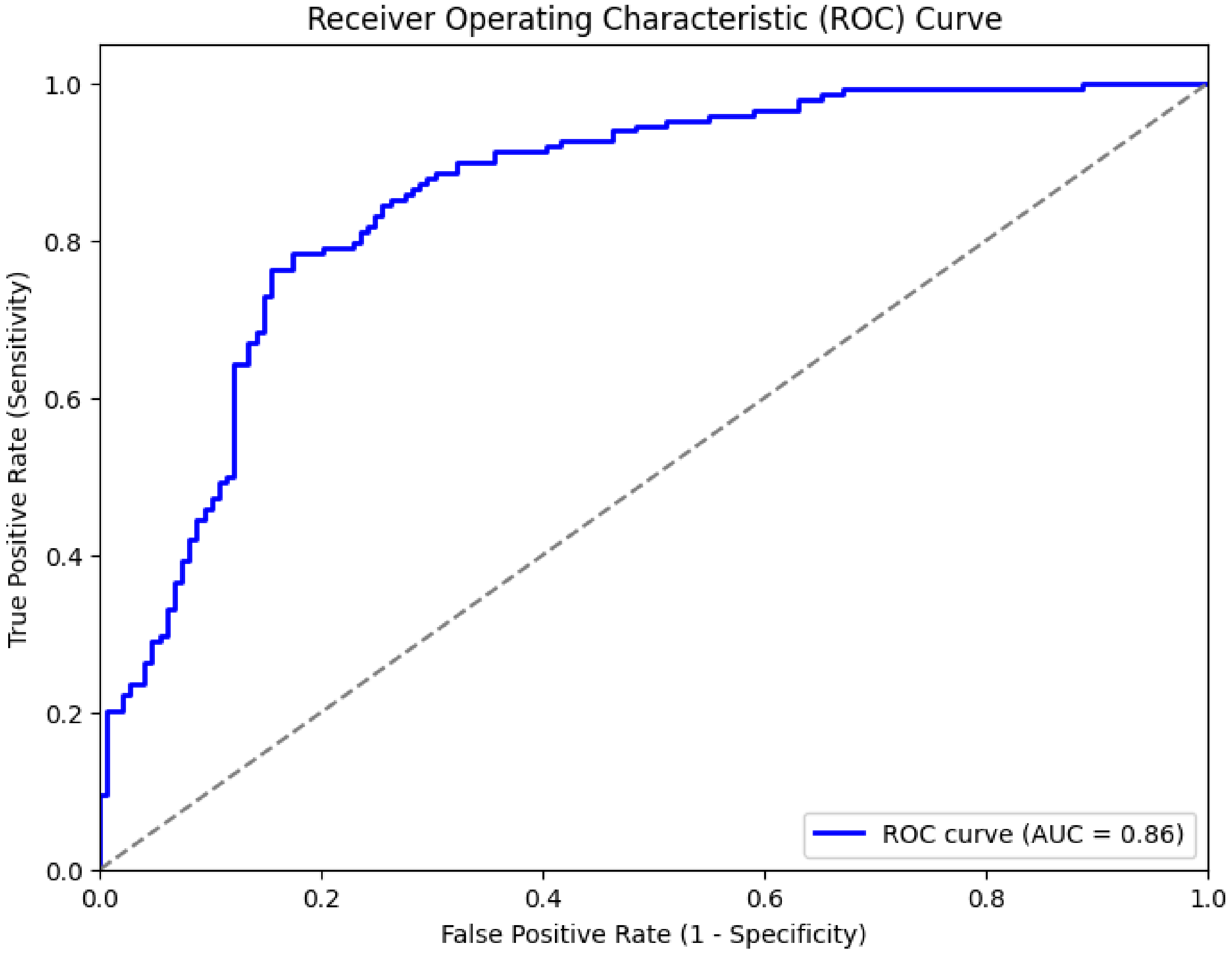
Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek


file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
```

```
plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.7941
  Fold 2: Accuracy = 0.7899
  Fold 3: Accuracy = 0.7553
  Fold 4: Accuracy = 0.8143
  Fold 5: Accuracy = 0.7764

Mean CV Accuracy: 0.7860

=== Final Test Set Evaluation ===
Accuracy: 0.7912
Precision: 0.7688
Recall: 0.8311
F1 Score: 0.7987
```

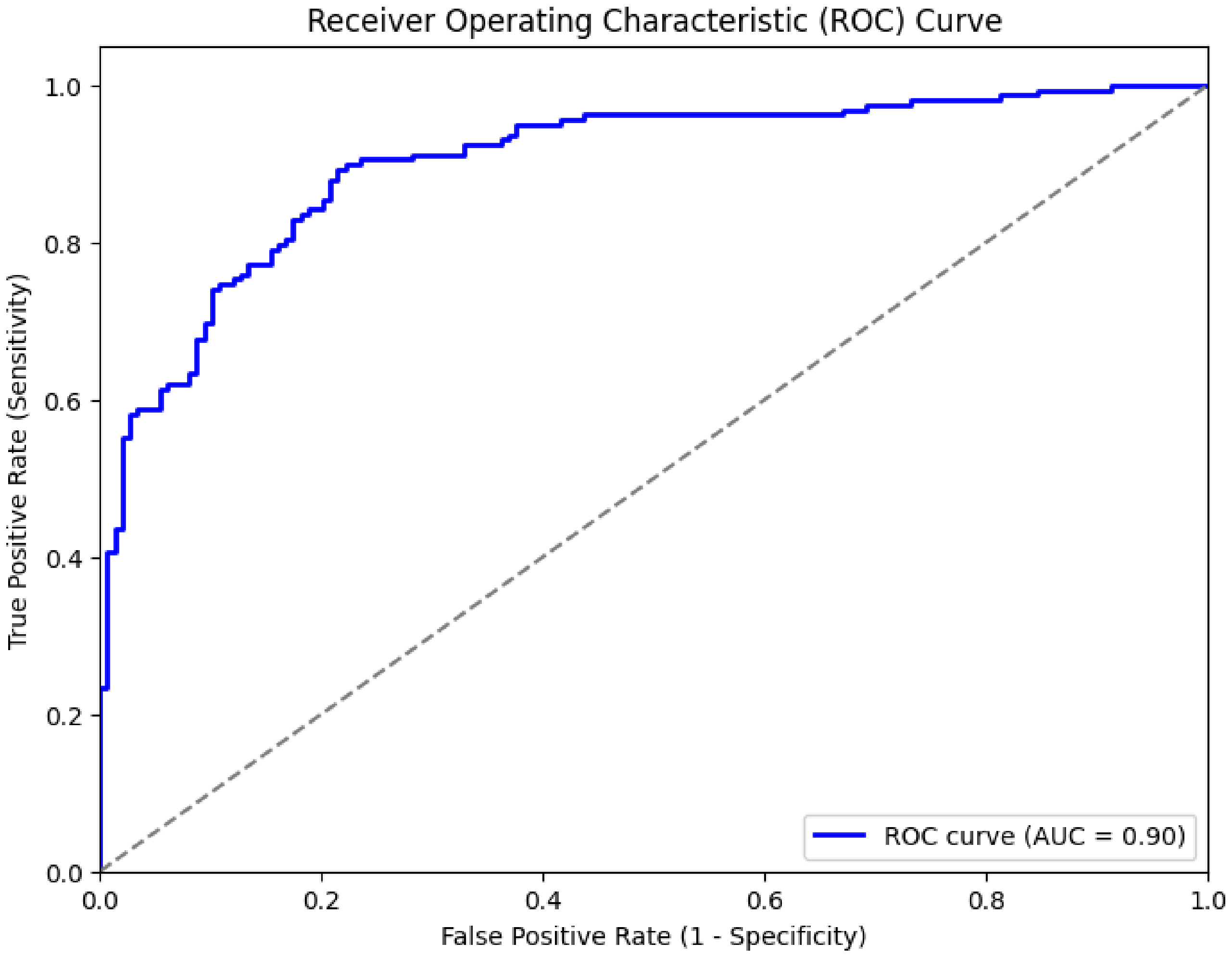


ADASYN - full model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import ADASYN

file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

adasyn = ADASYN(random_state=42)
X_balanced, y_balanced = adasyn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]

    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)
```

```
        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.8286
  Fold 2: Accuracy = 0.7469
  Fold 3: Accuracy = 0.7959
  Fold 4: Accuracy = 0.8041
  Fold 5: Accuracy = 0.7992

Mean CV Accuracy: 0.7949

=== Final Test Set Evaluation ===
Accuracy: 0.8143
Precision: 0.8389
Recall: 0.7911
F1 Score: 0.8143
```

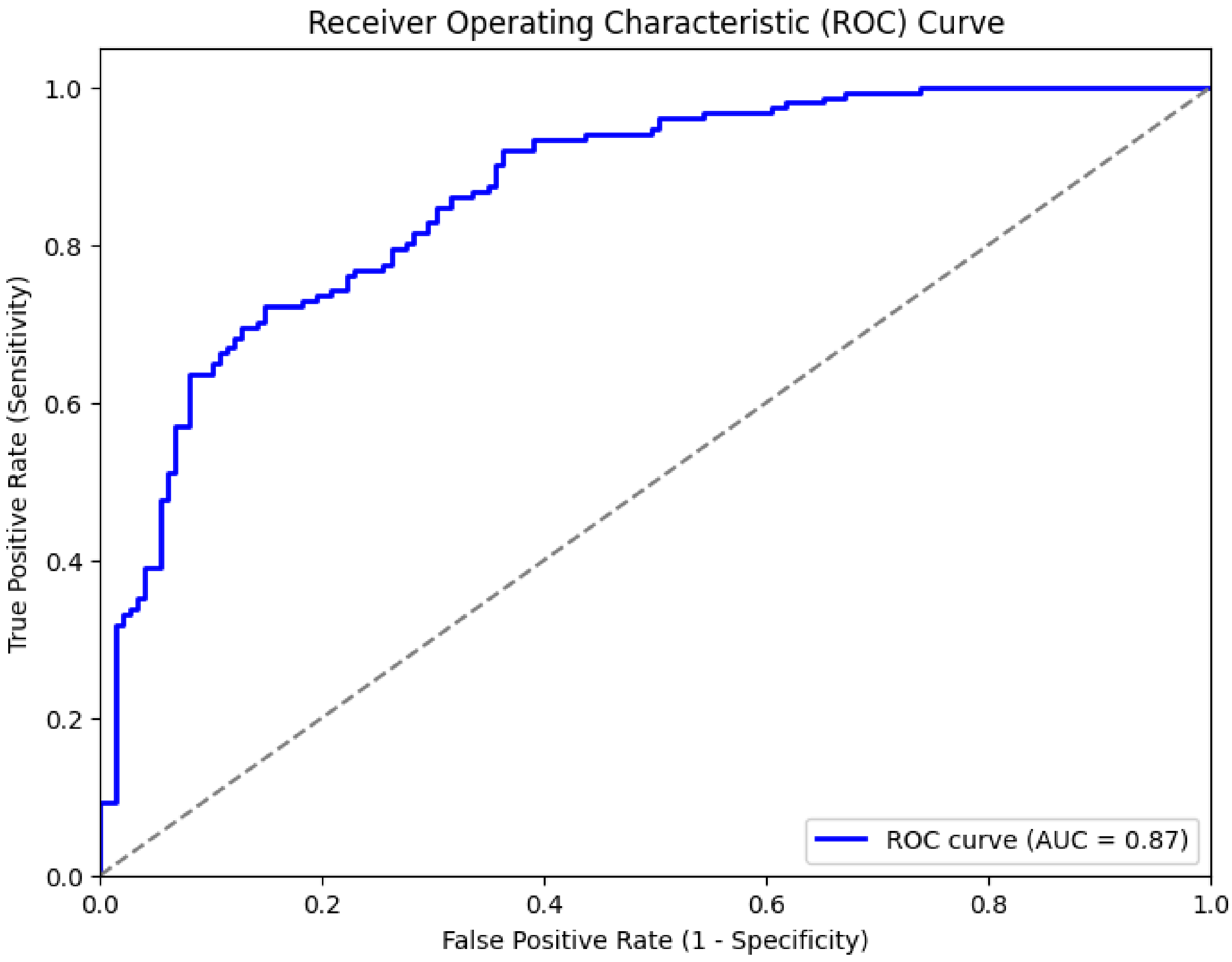


Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import ADASYN

file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

adasyn = ADASYN(random_state=42)
X_balanced, y_balanced = adasyn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]

    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)
```

```
        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.7375
  Fold 2: Accuracy = 0.7625
  Fold 3: Accuracy = 0.7875
  Fold 4: Accuracy = 0.7573
  Fold 5: Accuracy = 0.8410

Mean CV Accuracy: 0.7772

=== Final Test Set Evaluation ===
Accuracy: 0.7633
Precision: 0.7532
Recall: 0.7881
F1 Score: 0.7702
```

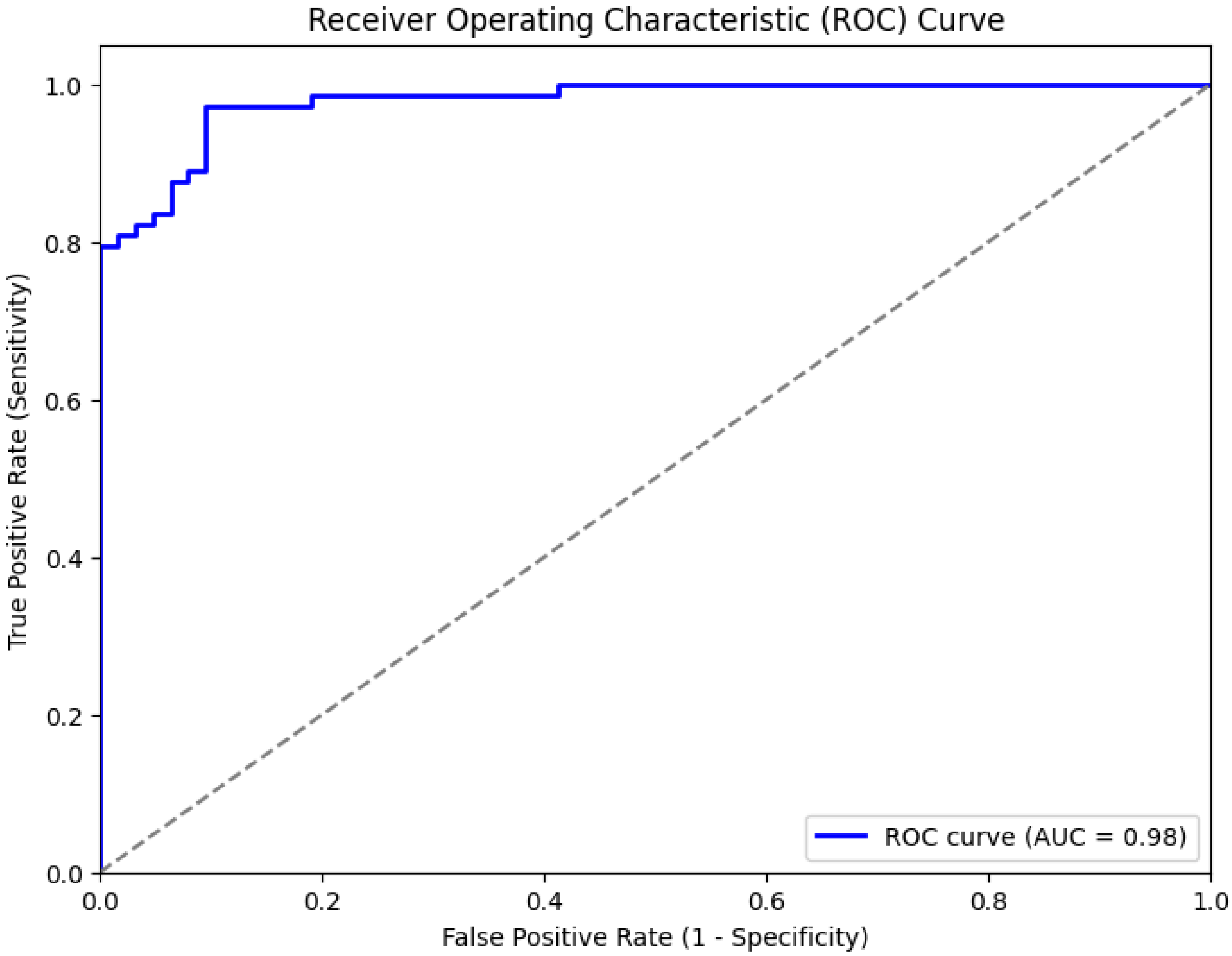


SMOTEEN - full model

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek


file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smoteenn = SMOTEENN(random_state=42)
X_balanced, y_balanced = smoteenn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")
    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)
    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]


    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.8981
  Fold 2: Accuracy = 0.9259
  Fold 3: Accuracy = 0.8519
  Fold 4: Accuracy = 0.9352
  Fold 5: Accuracy = 0.8889

Mean CV Accuracy: 0.9000

=== Final Test Set Evaluation ===
Accuracy: 0.9412
Precision: 0.9221
Recall: 0.9726
F1 Score: 0.9467
```

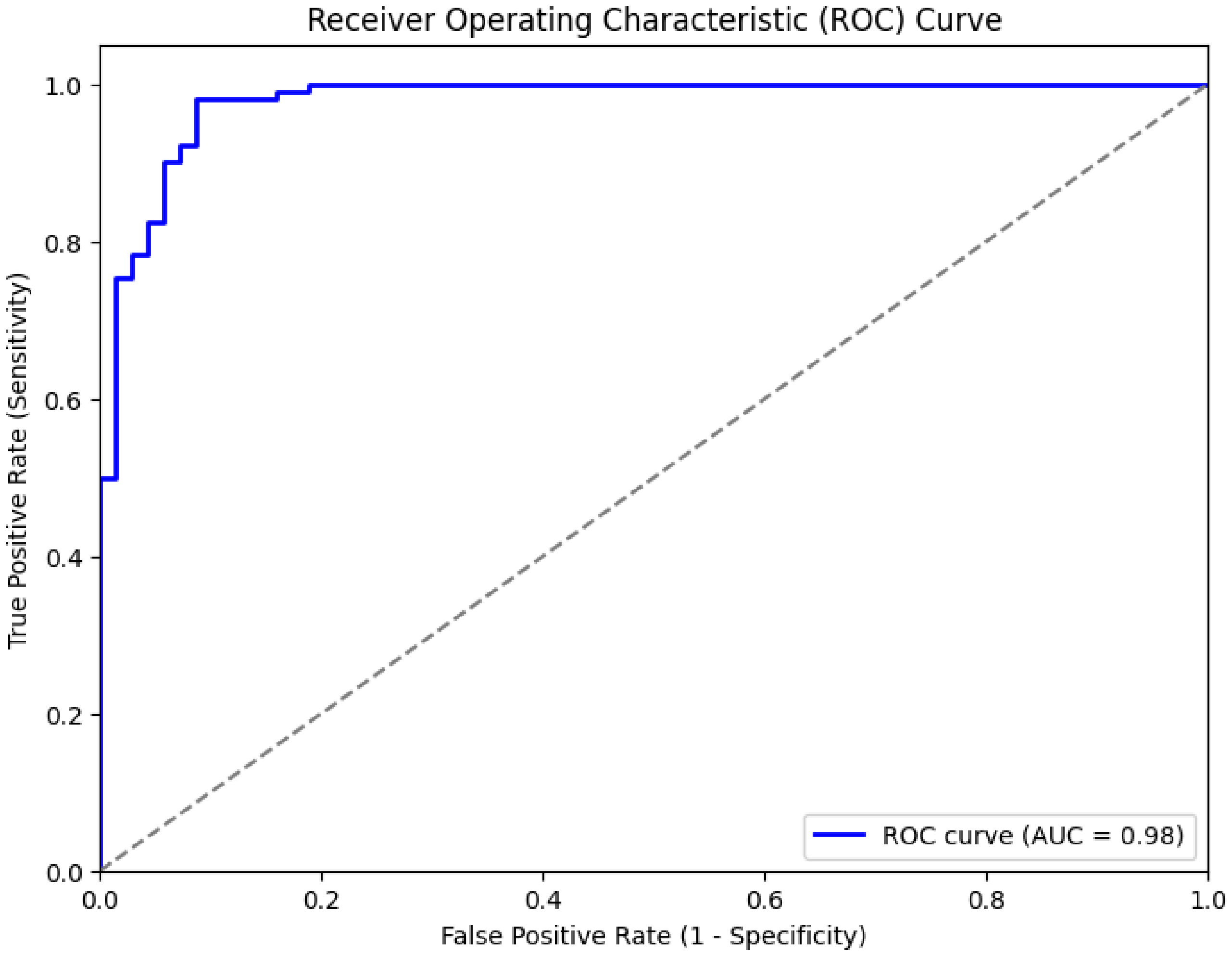


Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek


file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smoteenn = SMOTEENN(random_state=42)
X_balanced, y_balanced = smoteenn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")
    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)
    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]


    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.9343
  Fold 2: Accuracy = 0.9051
  Fold 3: Accuracy = 0.9270
  Fold 4: Accuracy = 0.9124
  Fold 5: Accuracy = 0.9412

Mean CV Accuracy: 0.9240

=== Final Test Set Evaluation ===
Accuracy: 0.9240
Precision: 0.8938
Recall: 0.9902
F1 Score: 0.9395
```



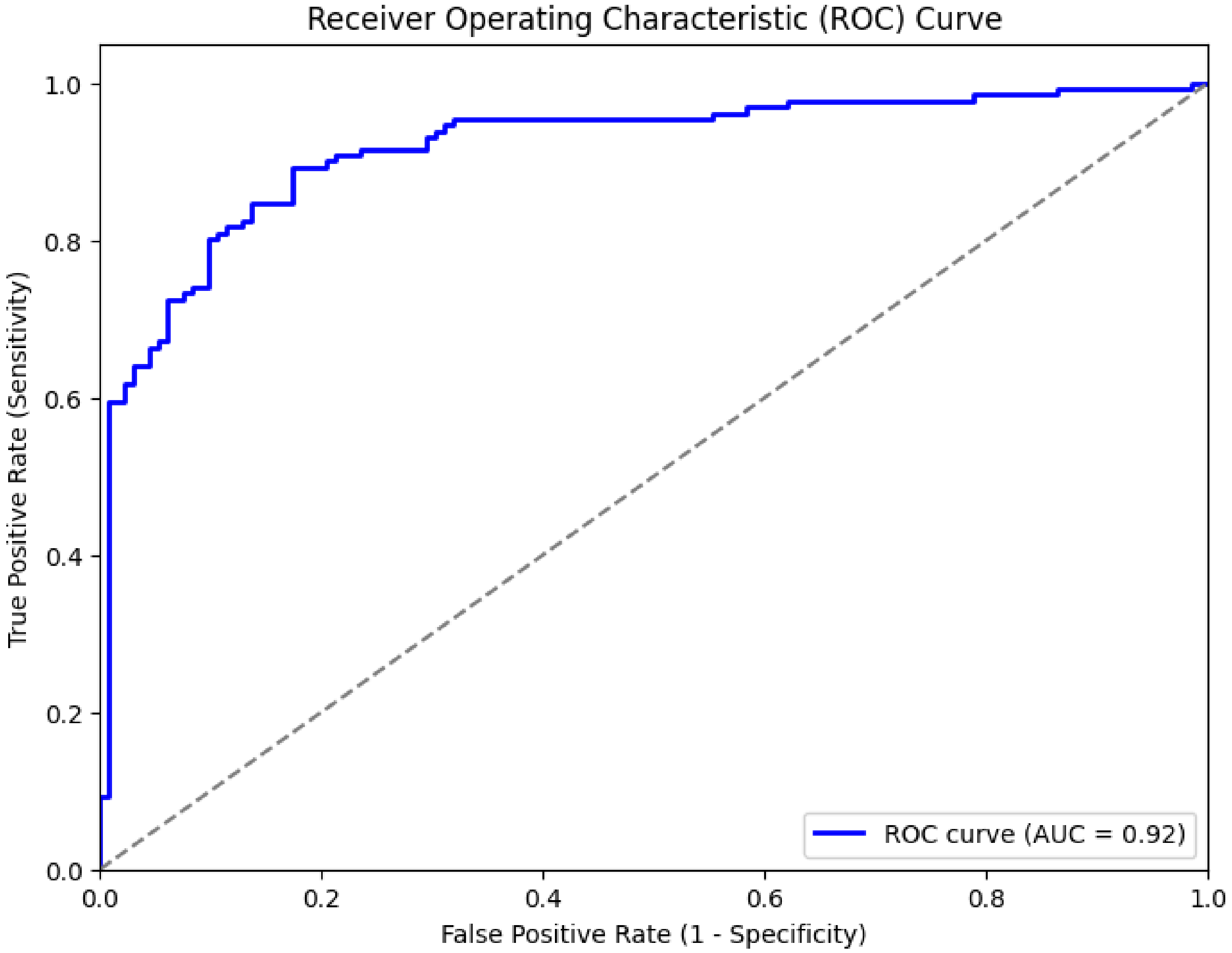
SMOTETomek - Full

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek

file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']



smotetomek = SMOTETomek(random_state=42)
X_balanced, y_balanced = smotetomek.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]



    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right')
plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.8341
  Fold 2: Accuracy = 0.8190
  Fold 3: Accuracy = 0.8333
  Fold 4: Accuracy = 0.8143
  Fold 5: Accuracy = 0.8048

Mean CV Accuracy: 0.8211

=== Final Test Set Evaluation ===
Accuracy: 0.8517
Precision: 0.8333
Recall: 0.8779
F1 Score: 0.8550
```



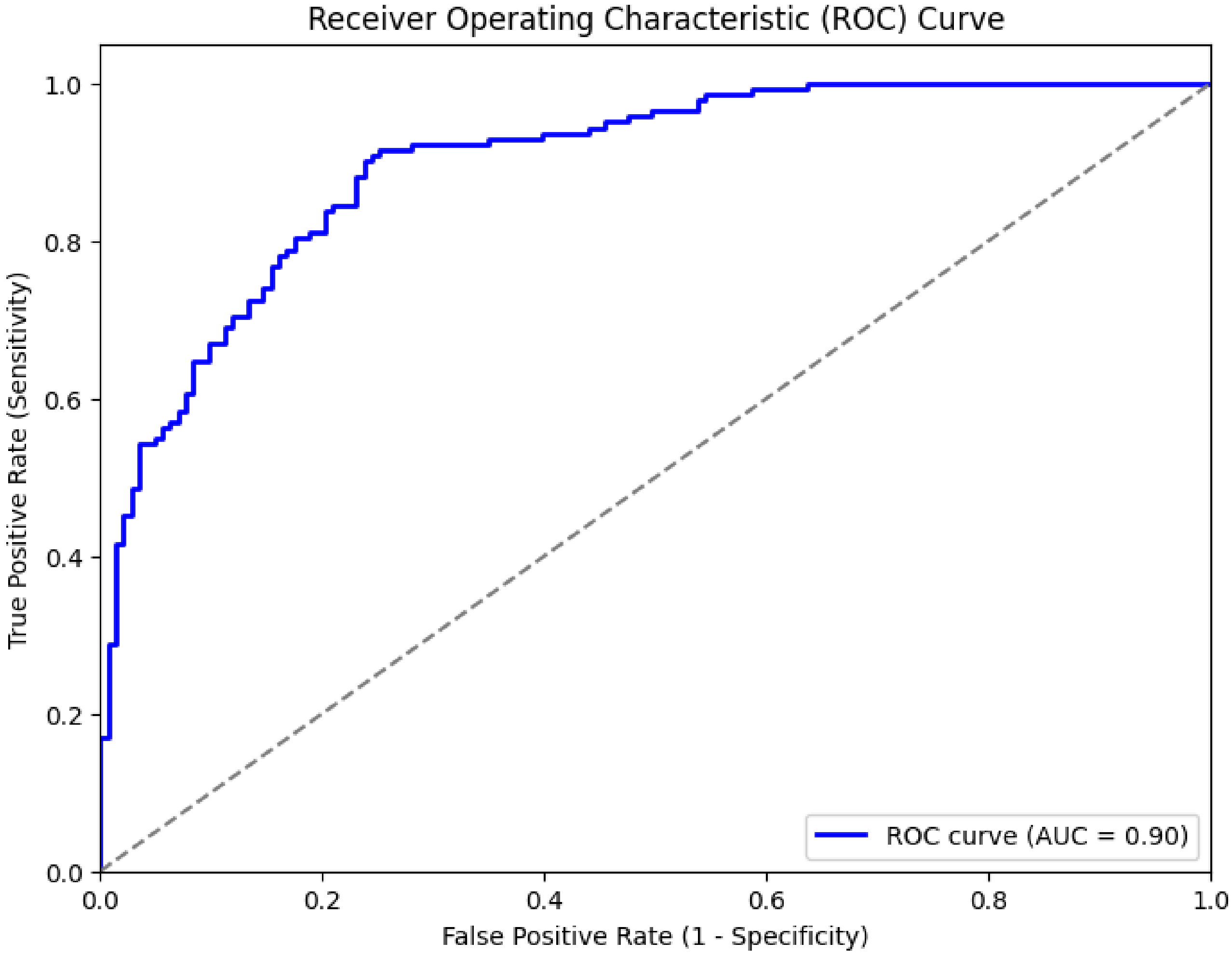
Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN
from imblearn.combine import SMOTETomek

file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']



smotetomek = SMOTETomek(random_state=42)
X_balanced, y_balanced = smotetomek.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = LogisticRegression(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]



    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
```

```
        meta_model.fit(meta_features, y_val_fold)

        y_pred_fold = meta_model.predict(meta_features)

        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
```

```
plt.legend(loc='lower right')
plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.8202
  Fold 2: Accuracy = 0.7939
  Fold 3: Accuracy = 0.7588
  Fold 4: Accuracy = 0.7851
  Fold 5: Accuracy = 0.8194

Mean CV Accuracy: 0.7955

=== Final Test Set Evaluation ===
Accuracy: 0.8140
Precision: 0.7947
Recall: 0.8451
F1 Score: 0.8191
```



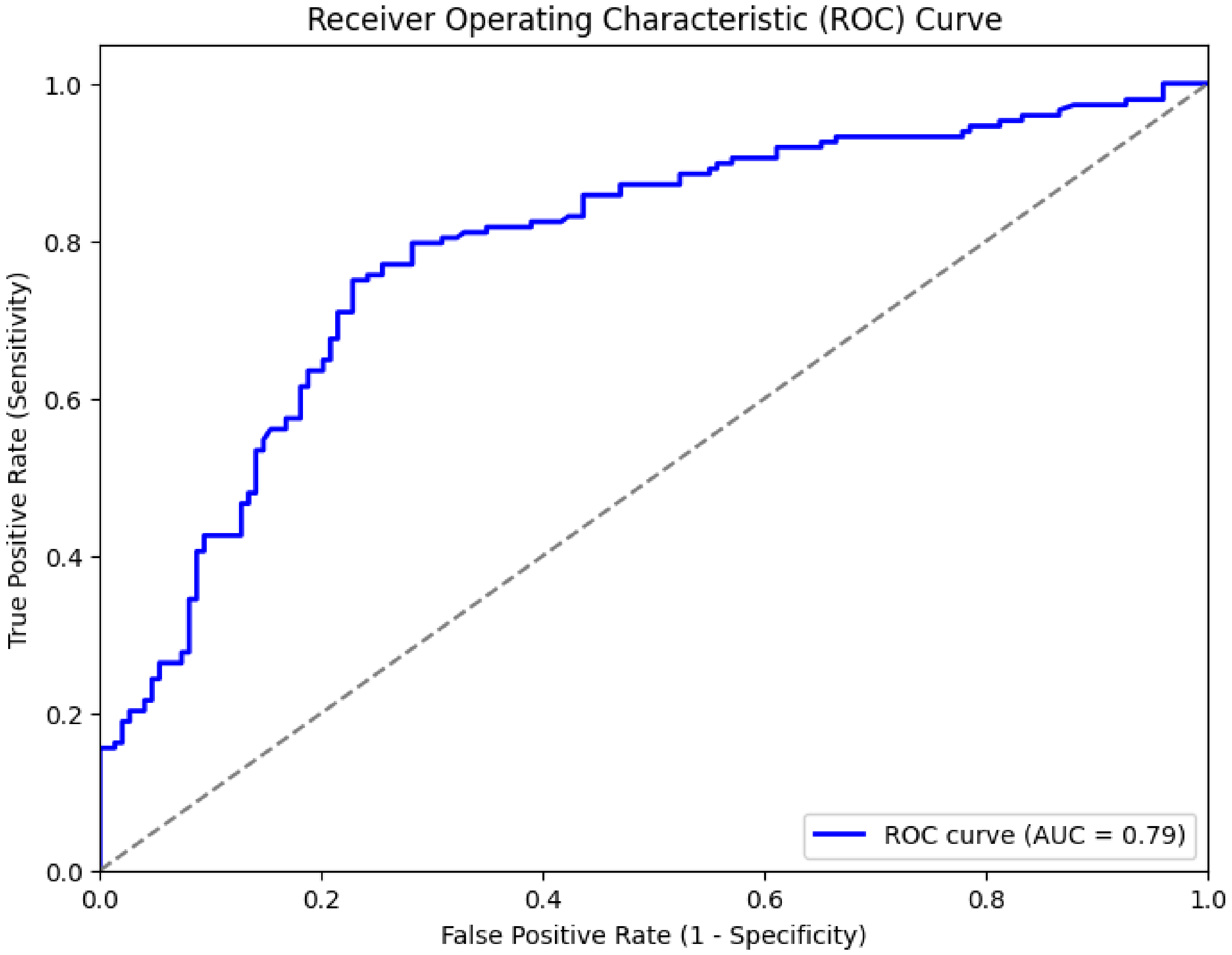
META in GB - Full model

SMOTE

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE

file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = GradientBoostingClassifier(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)

    accuracy = accuracy_score(y_val_fold, y_pred_fold)
```

```
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

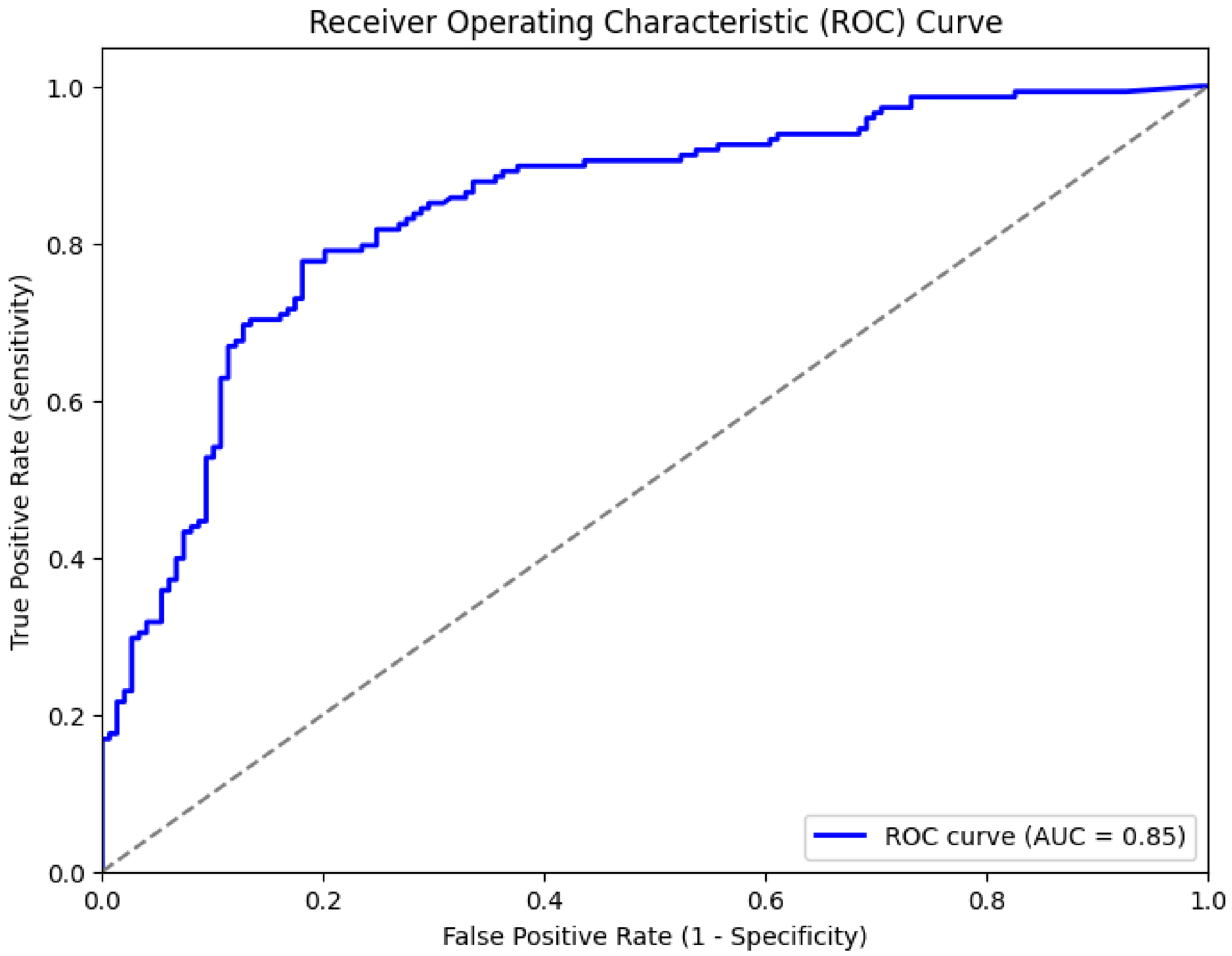
```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.9958
  Fold 2: Accuracy = 0.9958
  Fold 3: Accuracy = 0.9831
  Fold 4: Accuracy = 0.9958
  Fold 5: Accuracy = 0.9873

Mean CV Accuracy: 0.9916

=== Final Test Set Evaluation ===
Accuracy: 0.7475
Precision: 0.7589
Recall: 0.7230
F1 Score: 0.7405
```



Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import SMOTE

file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = GradientBoostingClassifier(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]
    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)

    accuracy = accuracy_score(y_val_fold, y_pred_fold)
    cv_scores.append(accuracy)
    print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

mean_cv_score = np.mean(cv_scores)
print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

print("\n=== Final Test Set Evaluation ===")

meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
for i, (model_name, model) in enumerate(base_models.items()):

    model.fit(X_train, y_train)

    meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

y_pred_test = meta_model.predict(meta_features_test)

accuracy = accuracy_score(y_test, y_pred_test)
precision = precision_score(y_test, y_pred_test)
recall = recall_score(y_test, y_pred_test)
f1 = f1_score(y_test, y_pred_test)

y_pred_test = meta_model.predict(meta_features_test)
y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

accuracy = accuracy_score(y_test, y_pred_test)
precision = precision_score(y_test, y_pred_test)
recall = recall_score(y_test, y_pred_test)
f1 = f1_score(y_test, y_pred_test)

print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")

from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.9916
  Fold 2: Accuracy = 1.0000
  Fold 3: Accuracy = 0.9831
  Fold 4: Accuracy = 1.0000
  Fold 5: Accuracy = 0.9831

Mean CV Accuracy: 0.9916

=== Final Test Set Evaluation ===
Accuracy: 0.7946
Precision: 0.7959
Recall: 0.7905
F1 Score: 0.7932
```

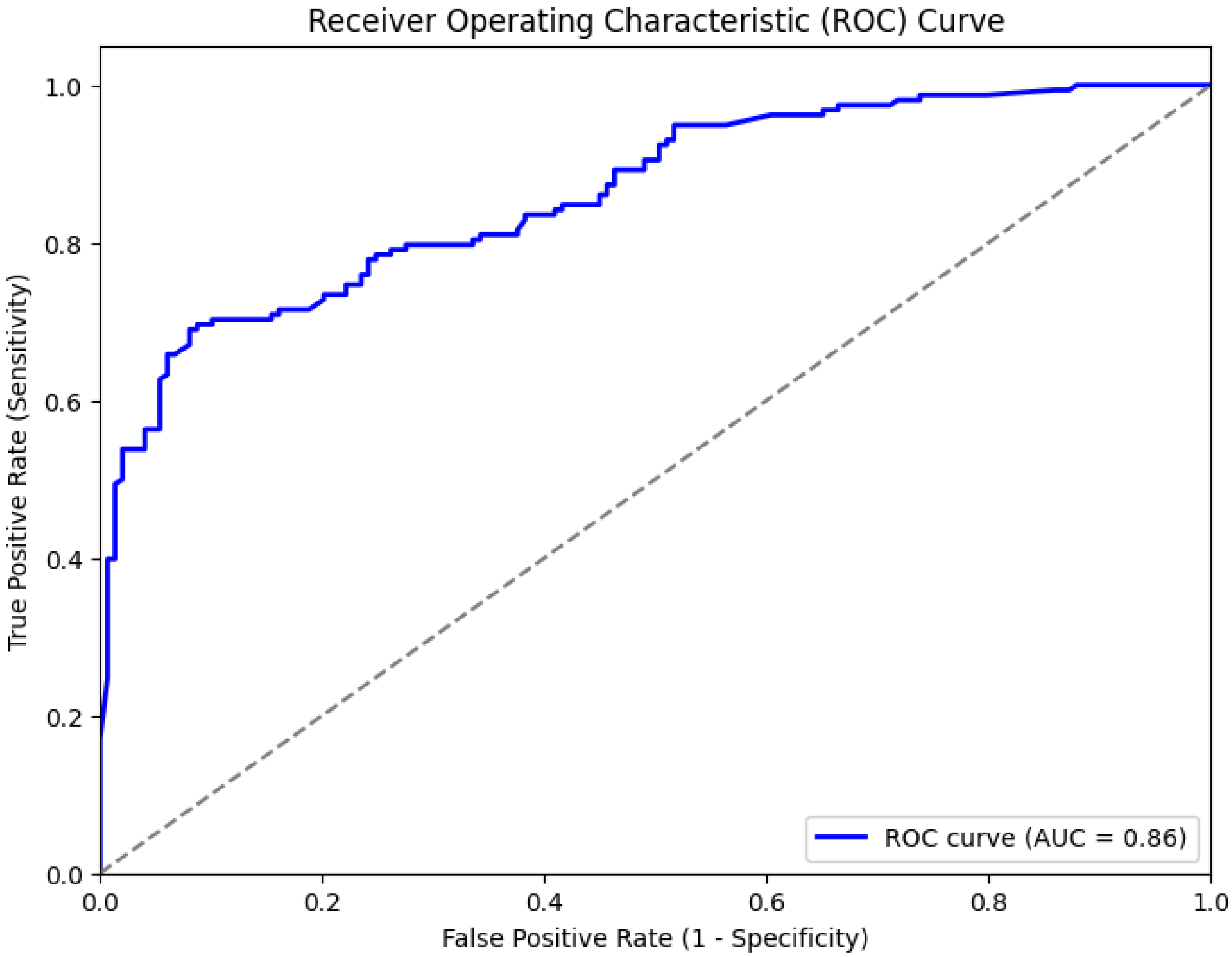


ADASYN - Full

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN

file_path = '/content/Data_Analysis_Jib.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

adasyn = ADASYN(random_state=42)
X_balanced, y_balanced = adasyn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = GradientBoostingClassifier(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]

    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)
```

```
        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 1.0000
  Fold 2: Accuracy = 0.9959
  Fold 3: Accuracy = 0.9959
  Fold 4: Accuracy = 0.9959
  Fold 5: Accuracy = 0.9836

Mean CV Accuracy: 0.9943

=== Final Test Set Evaluation ===
Accuracy: 0.7687
Precision: 0.7736
Recall: 0.7785
F1 Score: 0.7760
```

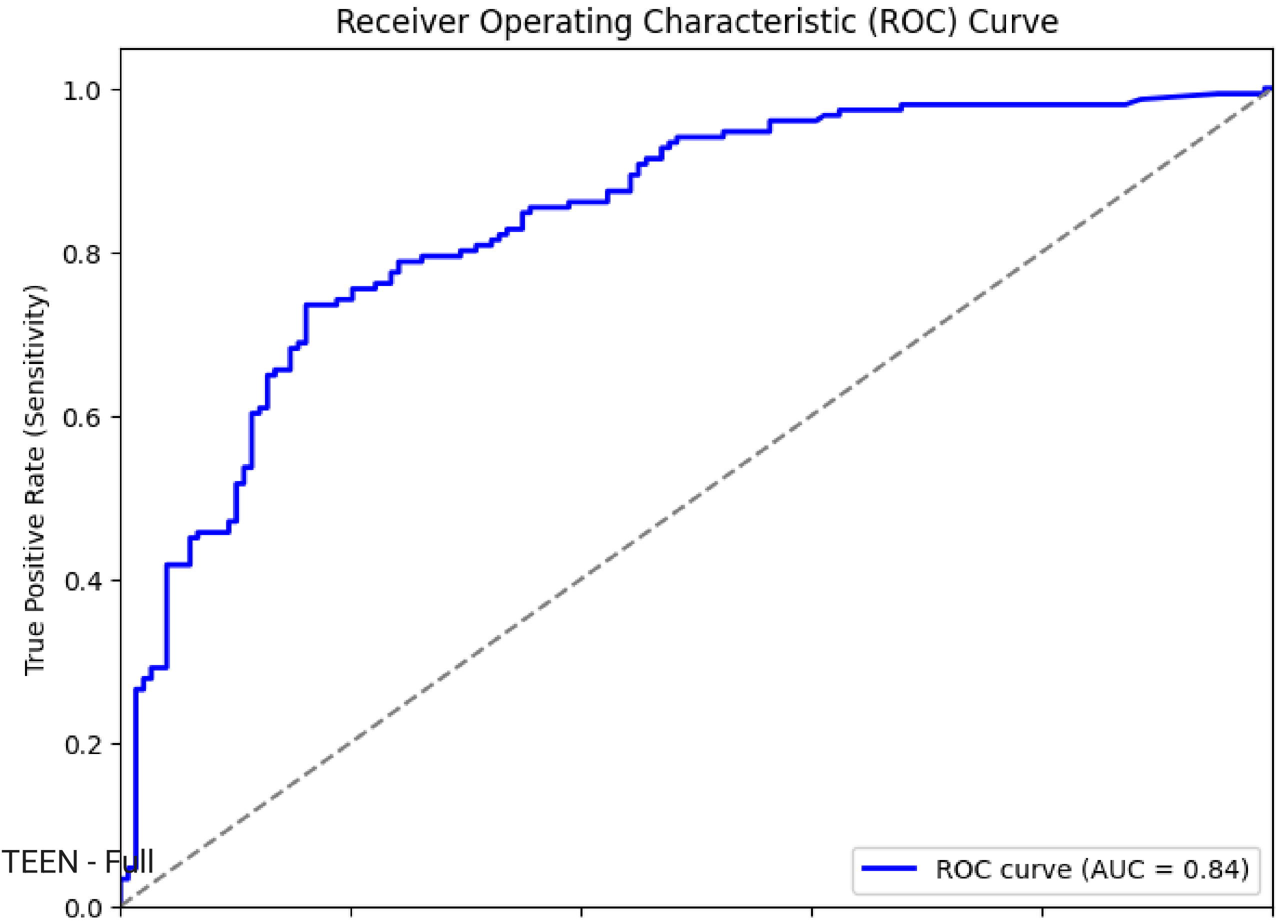


Stepwise

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from imblearn.over_sampling import ADASYN
from imblearn.combine import SMOTEENN

file_path = '/content/Data_Analysis_Jib - cut.xlsx'
data = pd.read_excel(file_path)

X = data.drop('Target', axis=1)
y = data['Target']

adasyn = ADASYN(random_state=42)
X_balanced, y_balanced = adasyn.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(
    X_balanced, y_balanced, test_size=0.2, random_state=42, stratify=y_balanced
)

base_models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(probability=True, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
}

meta_model = GradientBoostingClassifier(random_state=42)

kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = []

print("\n=== Two-Step Model: K-Fold Cross-Validation Process ===")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_train, y_train), 1):

    X_train_fold, X_val_fold = X_train.iloc[train_idx], X_train.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]

    meta_features = np.zeros((X_val_fold.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train_fold, y_train_fold)

        meta_features[:, i] = model.predict_proba(X_val_fold)[:, 1]

    meta_model.fit(meta_features, y_val_fold)

    y_pred_fold = meta_model.predict(meta_features)
```

```
        accuracy = accuracy_score(y_val_fold, y_pred_fold)
        cv_scores.append(accuracy)
        print(f"  Fold {fold}: Accuracy = {accuracy:.4f}")

    mean_cv_score = np.mean(cv_scores)
    print(f"\nMean CV Accuracy: {mean_cv_score:.4f}")

    print("\n=== Final Test Set Evaluation ===")

    meta_features_test = np.zeros((X_test.shape[0], len(base_models)))
    for i, (model_name, model) in enumerate(base_models.items()):

        model.fit(X_train, y_train)

        meta_features_test[:, i] = model.predict_proba(X_test)[:, 1]

    y_pred_test = meta_model.predict(meta_features_test)

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    y_pred_test = meta_model.predict(meta_features_test)
    y_pred_prob = meta_model.predict_proba(meta_features_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred_test)
    precision = precision_score(y_test, y_pred_test)
    recall = recall_score(y_test, y_pred_test)
    f1 = f1_score(y_test, y_pred_test)

    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")

    from sklearn.metrics import roc_curve, auc
    import matplotlib.pyplot as plt

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate (1 - Specificity)')
    plt.ylabel('True Positive Rate (Sensitivity)')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

```
=== Two-Step Model: K-Fold Cross-Validation Process ===
  Fold 1: Accuracy = 0.9875
  Fold 2: Accuracy = 0.9875
  Fold 3: Accuracy = 0.9708
  Fold 4: Accuracy = 0.9958
  Fold 5: Accuracy = 0.9916

Mean CV Accuracy: 0.9867

=== Final Test Set Evaluation ===
Accuracy: 0.7767
Precision: 0.8000
Recall: 0.7417
F1 Score: 0.7698
```



SMOTEEN - Full

```
import pandas as pd
import numpy as np
from sklearn.model_selection import StratifiedKFold, train_test_split
from sklearn.ensemble import GradientBoostingClassifier
```