**ROC**

df = pd.read\_csv('date resourse.csv')

all\_features = ['Sex', 'Age', 'BMI', 'CVH', 'Alcohol', 'Autoimmune', 'Cryptogenic', 'Ascites', 'HE', 'EGVH', 'Infection', 'Albumin', 'Hb', 'TBIL', 'Creatinine', 'INR', 'NLR', 'PLR', 'LMR', 'MELD', 'CTP', 'ALT', 'AST', 'ALP', 'GGT', 'LDH', 'PLT', 'Na']

y = df['Frail']

models = [

('Logit', LogisticRegression(max\_iter=10000, C=0.1), all\_features),

('KNN', KNeighborsClassifier(n\_neighbors=5), all\_features),

('ANN', MLPClassifier(hidden\_layer\_sizes=(100, 50), activation='relu', max\_iter=1000, random\_state=0), all\_features),

('RF', RandomForestClassifier(max\_depth=2, random\_state=0), all\_features),

('XGBoost', XGBClassifier(max\_depth=2, n\_estimators=100, learning\_rate=0.05, random\_state=42), all\_features)

]

n\_bootstraps = 500

roc\_data = {}

np.random.seed(42)

random.seed(42)

for name, model, features in models:

X = df[features]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model.fit(X\_train, y\_train)

for X\_set, y\_set, dataset in [(X\_train, y\_train, 'train'), (X\_test, y\_test, 'test')]:

probs = model.predict\_proba(X\_set)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_set, probs)

auc\_score = auc(fpr, tpr)

roc\_data[(name, dataset, 'AUC')] = auc\_score

plt.figure(figsize=(14, 7))

plt.subplot(1, 2, 1)

for name, model, features in models:

X = df[features]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

probs = model.predict\_proba(X\_train)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_train, probs)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'{name} (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curves Comparison on Training Set')

plt.legend(loc='lower right')

plt.subplot(1, 2, 2)

for name, model, features in models:

X = df[features]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

probs = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_test, probs)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f'{name} (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1]

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curves Comparison on Test Set')

plt.legend(loc='lower right')

plt.tight\_layout()

plt.show()

**Bar Plot and Beeswarm Plot**

df = pd.read\_csv(' date resourse.csv')

features = ['Sex','Age','BMI','CVH','Alcohol','Autoimmune','Cryptogenic','Ascites','HE','EGVH','Infection','Albumin'

,'Hb','TBIL','Creatinine','INR','NLR','PLR','LMR','MELD','CTP','ALT','AST','ALP','GGT','LDH','PLT','Na']

y = df['Frail']

X = df[features]

X\_train,X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

explainer = shap.TreeExplainer(model)

shap\_values = explainer.shap\_values(X\_test)

shap.summary\_plot(shap\_values[1], X\_test, plot\_type="bar")

shap.summary\_plot(shap\_values[1], X\_test, plot\_type="dot")

**Dependence Plot**

feature\_index = X\_test.columns.get\_loc('LMR')

feature\_values = X\_test['LMR']

shap.dependence\_plot('LMR', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='LMR')

plt.xlabel('LMR')

plt.ylabel('SHAP Value')

plt.show()

feature\_index = X\_test.columns.get\_loc('NLR')

feature\_values = X\_test['NLR']

shap.dependence\_plot('NLR', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='NLR')

plt.xlabel('NLR')

plt.ylabel('SHAP Value')

plt.show()

feature\_index = X\_test.columns.get\_loc('Creatinine')

feature\_values = X\_test['Creatinine']

shap.dependence\_plot('Creatinine', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='Creatinine')

plt.xlabel('Creatinine')

plt.ylabel('SHAP Value')

plt.show()

feature\_index = X\_test.columns.get\_loc('Ascites')

feature\_values = X\_test['Ascites']

shap.dependence\_plot('Ascites', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='Ascites')

plt.xlabel('Ascites')

plt.ylabel('SHAP Value')

plt.show()

feature\_index = X\_test.columns.get\_loc('Age')

feature\_values = X\_test['Age']

shap.dependence\_plot('Age', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='Age')

plt.xlabel('Age')

plt.ylabel('SHAP Value')

plt.show()

feature\_index = X\_test.columns.get\_loc('Albumin')

feature\_values = X\_test['Albumin']

shap.dependence\_plot('Albumin', shap\_values[1], X\_test, interaction\_index=None, show=False)

plt.scatter(feature\_values, shap\_values[1][:, feature\_index], c=feature\_values, cmap='coolwarm')

plt.axhline(0, color='red', linestyle='--', label='SHAP Value = 0')

plt.colorbar(label='Albumin')

plt.xlabel('Albumin')

plt.ylabel('SHAP Value')

plt.show()

**Force Plot**

shap\_values\_instance = explainer.shap\_values(X\_train.iloc[38])

shap.force\_plot(explainer.expected\_value[1], shap\_values\_instance[1], X\_train.iloc[38], matplotlib=True)

y\_pred\_proba = model.predict\_proba(X\_train.iloc[[38]])

y\_pred = model.predict(X\_train.iloc[[38]])

print(y\_pred\_proba[0][1])

print(y\_pred[0])

**Decision Plot**

explainer = shap.TreeExplainer(model)

expected\_value = explainer.expected\_value[1]

features = X\_train.iloc[range(100)]

shap\_values = explainer.shap\_values(features)[1]

feature\_sample = features.iloc[38]

shap\_values\_sample = shap\_values[38]

shap.decision\_plot(expected\_value, shap\_values\_sample, feature\_sample)