```python
### 橫斷研究模型
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, auc, accuracy_score,
recall_score, precision_score, f1_score
from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from imblearn.over_sampling import SMOTE

# 1. 讀取資料
df_raw = pd.read_csv('oasis_cross-sectional.csv')

# 2. 視覺化分析 (熱力圖)
heatmap_cols = ['CDR', 'Age', 'Educ', 'SES', 'MMSE', 'nWBV',
'eTIV', 'ASF']
plt.figure(figsize=(10, 8))
corr_matrix = df_raw[heatmap_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
fmt=".2f", linewidths=0.5, vmin=-1, vmax=1)
plt.title('Correlation Heatmap (Raw Data)', fontsize=14)
plt.show()

# 3. 資料前處理
# 排除重複測量
df = df_raw[df_raw['Delay'].isnull()].copy()

# 定義目標變數 Y 、性別轉成數字
df['Dementia'] = df['CDR'].apply(lambda x: 1 if x > 0 else 0)
df['M/F'] = df['M/F'].apply(lambda x: 1 if x == 'M' else 0)

# 異常值檢查
check_cols = ['Age', 'nWBV', 'MMSE', 'eTIV', 'Educ', 'ASF']
for col in check_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    outliers = df[(df[col] < lower) | (df[col] > upper)]

# 遺漏值處理
# A: 補值
df_imp = df.copy()
# 用中位數
df_imp['SES'] = df_imp['SES'].fillna(df_imp['SES'].median())
df_imp['MMSE'] =
df_imp['MMSE'].fillna(df_imp['MMSE'].median())
df_imp['Educ'] =
df_imp['Educ'].fillna(df_imp['Educ'].median())

features_A = ['M/F', 'Age', 'Educ', 'SES', 'MMSE', 'nWBV']
X_imp = df_imp[features_A].values
y_imp = df_imp['Dementia'].values

# B: 刪除變項
features_B = ['M/F', 'Age', 'nWBV']
X_drop = df[features_B].values
y_drop = df['Dementia'].values

# 4. 實驗設計（資料拆分：5-Fold CV、SMOTE）、訓練模型

def train_and_evaluate(model_class, X, y, model_name,
n_splits=5):
    skf = StratifiedKFold(n_splits=n_splits, shuffle=True,
random_state=42)
    metrics_sum = {'Accuracy': 0, 'Precision': 0, 'Recall':
0, 'F1': 0}
    tprs = []
    base_fpr = np.linspace(0, 1, 101)
    fold_details = []

    print(f"\n[{model_name}] 訓練詳細數據:")
    print(f"{'Fold':<6} {'原始Train (0/1)':<16} {'SMOTE後Train
(0/1)':<16} {'Test (0/1)':<14} {'Acc':<8} {'Rec':<8}
{'Pre':<8} {'F1':<8}")
    print("-" * 100)

    fold_idx = 1
    for train_idx, val_idx in skf.split(X, y):
        X_train, X_val = X[train_idx], X[val_idx]
        y_train, y_val = y[train_idx], y[val_idx]
        orig_0, orig_1 = np.bincount(y_train)

        smote = SMOTE(random_state=42)
        X_train_res, y_train_res =
smote.fit_resample(X_train, y_train)
        res_0, res_1 = np.bincount(y_train_res)
        # 訓練模型
        model = model_class(random_state=42)
        model.fit(X_train_res, y_train_res)
        y_pred = model.predict(X_val)
        y_prob = model.predict_proba(X_val)[:, 1]
        acc = accuracy_score(y_val, y_pred)
        pre = precision_score(y_val, y_pred, zero_division=0)
        rec = recall_score(y_val, y_pred)
        f1 = f1_score(y_val, y_pred)
        metrics_sum['Accuracy'] += acc
        metrics_sum['Precision'] += pre
        metrics_sum['Recall'] += rec
        metrics_sum['F1'] += f1

        # ROC 曲線
        fpr, tpr, _ = roc_curve(y_val, y_prob)
        interp_tpr = np.interp(base_fpr, fpr, tpr)
        interp_tpr[0] = 0.0
        tprs.append(interp_tpr)
        test_0, test_1 = np.bincount(y_val)
        print(f"{fold_idx:<6} {f'{orig_0}/{orig_1}':<16}
{f'{res_0}/{res_1}':<16} {f'{test_0}/{test_1}':<14} {acc:.3f}
{rec:.3f}    {pre:.3f}    {f1:.3f}")

        fold_details.append({'fold': fold_idx, 'fpr': fpr,
'tpr': tpr, 'auc': auc(fpr, tpr)})
        fold_idx += 1

    # 計算平均指標
    avg_metrics = {k: v / n_splits for k, v in
metrics_sum.items()}
    avg_metrics['Model'] = model_name

    mean_tpr = np.mean(tprs, axis=0)
    mean_tpr[-1] = 1.0
    mean_auc = auc(base_fpr, mean_tpr)

    return avg_metrics, base_fpr, mean_tpr, mean_auc,
fold_details

results = []
roc_data = []

# 跑隨機森林（補值）
res, base_fpr, mean_tpr, mean_auc, details =
train_and_evaluate(
    lambda random_state:
RandomForestClassifier(n_estimators=100,
random_state=random_state),
    X_imp, y_imp, 'RF (補值策略)'
)
results.append(res)
roc_data.append({'name': 'RF(imputed)', 'base_fpr': base_fpr,
'mean_tpr': mean_tpr, 'mean_auc': mean_auc, 'details':
details})

# 跑隨機森林 (刪除變項)
res, base_fpr, mean_tpr, mean_auc, details =
train_and_evaluate(
    lambda random_state:
RandomForestClassifier(n_estimators=100,
random_state=random_state),
    X_drop, y_drop, 'RF (刪除策略)'
)
results.append(res)
roc_data.append({'name': 'RF(drop)', 'base_fpr': base_fpr,
'mean_tpr': mean_tpr, 'mean_auc': mean_auc, 'details':
details})

# 6. 畫 ROC 圖
fig, axes = plt.subplots(1, 2, figsize=(16, 7))

for i, data in enumerate(roc_data):
    ax = axes[i]
    name = data['name']
    details = data['details']

    for d in details:
        ax.plot(d['fpr'], d['tpr'], lw=1.5, alpha=0.3,
label=f"Fold {d['fold']} (AUC = {d['auc']:.2f})")

    ax.plot(data['base_fpr'], data['mean_tpr'], color='b',
lw=2, label=f"Mean ROC (AUC = {data['mean_auc']:.2f})")
    ax.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r',
alpha=0.8)

    ax.set_title(f'{name} - ROC Curve', fontsize=14)
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.legend(loc="lower right")
    ax.grid(alpha=0.3)

plt.tight_layout()
plt.show()

# 7. 最終平均結果
df_res = pd.DataFrame(results).drop(columns=['Model'])
df_res.index = [r['Model'] for r in results]
print(df_res.round(3))


### 橫斷研究SHAP值
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import shap
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
# 訓練模型
df_raw = pd.read_csv('oasis_cross-sectional.csv')
df = df_raw[df_raw['Delay'].isnull()].copy()
df['Dementia'] = df['CDR'].apply(lambda x: 1 if x > 0 else 0)
df['M/F'] = df['M/F'].apply(lambda x: 1 if x == 'M' else 0)

# 策略 A: 補值
df_imp = df.copy()
df_imp['SES'] = df_imp['SES'].fillna(df_imp['SES'].median())
df_imp['MMSE'] =
df_imp['MMSE'].fillna(df_imp['MMSE'].median())
df_imp['Educ'] =
df_imp['Educ'].fillna(df_imp['Educ'].median())

features_A = ['M/F', 'Age', 'Educ', 'SES', 'MMSE', 'nWBV']
X = df_imp[features_A]
y = df_imp['Dementia']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)

model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# 計算SHAP
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
shap_vals_dementia = shap_values[:, :, 1]

# 生圖
plt.figure(figsize=(10, 6))
plt.title("SHAP Summary Plot (Impact on Risk)", fontsize=16)
shap.plots.beeswarm(shap_vals_dementia, show=False)
plt.tight_layout()
plt.show()


### 縱向研究模型
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score

# df_pair (baseline + followup)
if "df_pair" not in globals():
    df_long = pd.read_csv("oasis_longitudinal.csv").copy()
    df_long = df_long.sort_values(["Subject ID", "Visit"])
    base = df_long.groupby("Subject ID",
as_index=False).first()
    foll = df_long.groupby("Subject ID",
as_index=False).last()
    df_pair = base.merge(foll, on="Subject ID",
suffixes=("_baseline", "_followup"))

def pick_col(cands, cols):
    return next((c for c in cands if c in cols), None)

cols = df_pair.columns

# y：未來nWBV
y_col = pick_col(["nWBV_followup"], cols)

age_b  = pick_col(["Age_baseline"], cols)
mmse_b = pick_col(["MMSE_baseline"], cols)
cdr_b  = pick_col(["CDR_baseline"], cols)
nwbv_b = pick_col(["nWBV_baseline"], cols)
mr_b   = pick_col(["MR Delay_baseline", "Delay_baseline"],
cols)
mr_f   = pick_col(["MR Delay_followup", "Delay_followup"],
cols)
need = {"Age_baseline": age_b, "MMSE_baseline": mmse_b,
"CDR_baseline": cdr_b, "nWBV_baseline": nwbv_b}
df_pair = df_pair.copy()
```

```python
df_pair = df_pair[df_pair["followup_days"].notna() &
(df_pair["followup_days"] > 0)].copy()

# 選擇：X
X_cols = []
X_cols += [age_b, mmse_b, cdr_b, nwbv_b, "followup_days"]
X = df_pair[X_cols].copy()
y = df_pair[y_col].copy()
data = pd.concat([X, y.rename("y")], axis=1).dropna()
X = data[X_cols]
y = data["y"]


# 訓練/測試
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

gbr = GradientBoostingRegressor(random_state=42)
gbr.fit(X_train, y_train)
pred = gbr.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, pred))
mae  = mean_absolute_error(y_test, pred)
r2   = r2_score(y_test, pred)

print("RMSE:", rmse)
print("MAE :", mae)
print("R2  :", r2)


# 特徵重要度
imp = pd.Series(gbr.feature_importances_,
index=X_cols).sort_values(ascending=False)
print("\nFeature importances:")
print(imp)


### 縱向研究散布圖
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GroupKFold
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from scipy.stats import pearsonr
```

```python
# 資料前處理
if 'df_long' not in globals():
    df_long = pd.read_csv("oasis_longitudinal.csv")

df_long = df_long.sort_values(["Subject ID", "Visit"])
base = df_long.groupby("Subject ID", as_index=False).first()
foll = df_long.groupby("Subject ID", as_index=False).last()
df_pair = base.merge(foll, on="Subject ID",
suffixes=("_baseline", "_followup"))
df_pair["followup_days"] = df_pair["MR Delay_followup"]
df_pair = df_pair[df_pair["followup_days"].notna() &
(df_pair["followup_days"] > 0)].copy()

X_cols = ["Age_baseline", "MMSE_baseline", "CDR_baseline",
"nWBV_baseline", "followup_days"]
y_col = "nWBV_followup"

# 移除缺失值
df_pair = df_pair.dropna(subset=X_cols + [y_col])
X = df_pair[X_cols]
y = df_pair[y_col]
groups = df_pair["Subject ID"].values

# 執行收集預測值
gbr = GradientBoostingRegressor(random_state=42)
cv = GroupKFold(n_splits=5)

y_true_all = []
y_pred_all = []

for train_idx, test_idx in cv.split(X, y, groups=groups):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    gbr.fit(X_train, y_train)
    pred = gbr.predict(X_test)

    y_true_all.extend(y_test)
    y_pred_all.extend(pred)

# 轉換成 numpy array
y_true_all = np.array(y_true_all)
y_pred_all = np.array(y_pred_all)
overall_r2 = r2_score(y_true_all, y_pred_all)
overall_rmse = np.sqrt(mean_squared_error(y_true_all,
y_pred_all))
```

```
corr, p_value = pearsonr(y_true_all, y_pred_all)

# 畫圖
plt.figure(figsize=(8, 8))
plt.scatter(y_true_all, y_pred_all, alpha=0.6,
color='#4A90E2', edgecolors='white', s=60, label='Predicted
Points')
min_val = min(y_true_all.min(), y_pred_all.min())
max_val = max(y_true_all.max(), y_pred_all.max())
plt.plot([min_val, max_val], [min_val, max_val], 'k--', lw=2,
label='Ideal (y=x)')
m, b = np.polyfit(y_true_all, y_pred_all, 1)
plt.plot(y_true_all, m*y_true_all + b, color='red',
linestyle='--', lw=2, label=f'Fit Line')
plt.title(f"Predicted vs Actual (5-Fold CV)
\nR2={overall_r2:.3f}, RMSE={overall_rmse:.3f}, Pearson
r={corr:.3f}", fontsize=14, fontweight='bold')
plt.xlabel("Actual Follow-up nWBV", fontsize=12)
plt.ylabel("Predicted Follow-up nWBV", fontsize=12)
ax = plt.gca()
for spine in ax.spines.values():
    spine.set_edgecolor('black')
    spine.set_linewidth(1.5)

plt.legend(loc='upper left', frameon=True, framealpha=0.9)
plt.grid(True, linestyle=':', alpha=0.6)
plt.tight_layout()

plt.show()
```