In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import StratifiedKFold, KFold, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score
from sklearn.preprocessing import StandardScaler

import shap
import warnings
warnings.filterwarnings('ignore')
```

```
c:\Users\vighn\anaconda3\envs\helix_env\lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
 from .autonotebook import tqdm as notebook_tqdm
```

In [3]:
```python
# Paths
omics_dir = "OMICS DATA/"

# Load exposure and omics risk score files
exposure_df = pd.read_csv("helix_extracted.csv")
methylome = pd.read_csv(omics_dir + "methylome_risk_scores.csv")
transcriptome = pd.read_csv(omics_dir + "transcript_risk_scores.csv")
mirna = pd.read_csv(omics_dir + "mirna_risk_scores.csv")
proteome = pd.read_csv(omics_dir + "protein_risk_scores.csv")
metabolome = pd.read_csv(omics_dir + "metabolite_risk_scores.csv")
risk_weighted = pd.read_csv(omics_dir + "RISK_weighted.csv")
```

In [4]:
```python
# Base dataframe
master_df = exposure_df.copy()

# Merge all omics and final risk score
dfs = [methylome, transcriptome, mirna, proteome, metabolome, risk_weighted]
for df in dfs:
    master_df = master_df.merge(df, on="ID", how="inner")

print("Final merged shape:", master_df.shape)
master_df.head()
```

```
Final merged shape: (1301, 255)
```

Out[4]:

	ID	h_abs_ratio_preg_Log	h_no2_ratio_preg_Log	h_pm10_ratio_preg_None	h_pm25_ratio_preg_None	hs_no2_dy_hs_h_Log	hs_no2_wk_hs_h_Log	hs_no2_yr_hs_h_Log	hs_pm10_dy_hs_h_None	hs_pm10_wk_hs_h_None	...	protein_pregnancy_risk_y	total_protein_risk_y	trans
0	1	0.896711	2.872304	25.948498	17.433798	2.530279	2.583284	2.612098	22.535828	20.850005	...	-1.078877	-19.226702	
1	2	0.892538	2.980008	25.897739	18.470850	1.928600	2.652479	2.761064	14.077763	29.141274	...	-1.200788	-2.392757	
2	3	0.778723	3.056501	26.087347	18.711547	2.882591	2.591756	2.356163	46.859096	31.530981	...	-2.124728	-15.881561	
3	4	0.089056	3.089157	14.991380	16.409771	1.390750	2.456717	2.403247	29.817442	25.232778	...	-1.423351	-21.826047	
4	5	0.604781	3.848211	35.197296	14.889958	3.204449	3.499594	3.307663	29.817442	24.891465	...	-0.891981	-36.868356	

5 rows × 255 columns

In [5]:
```python
# Continuous outcome
y_cont = master_df["risk_score"].copy()

# Binary outcome: high risk = top 25%
threshold = np.percentile(y_cont, 75)
y_bin = (y_cont >= threshold).astype(int)

print("Risk threshold for classification:", round(threshold, 3))
print("Positive class proportion:", y_bin.mean())
```

```
Risk threshold for classification: 67.31
Positive class proportion: 0.25057647963105306
```

In [6]:
```python
omics_cols = [
    'total_methylome_risk_y',
    'total_transcript_risk_y',
    'total_mirna_risk_y',
    'total_protein_risk_y',
    'total_metabo_risk_y'
]

X_omics = master_df[omics_cols].copy()
X_omics.head()
```

Out[6]:

	total_methylome_risk_y	total_transcript_risk_y	total_mirna_risk_y	total_protein_risk_y	total_metabo_risk_y
0	-0.389442	-43.140855	3.523050	-19.226702	-123.855325
1	-1.802544	13.331690	4.447431	-2.392757	-64.614257
2	-9.394559	-12.763667	1.017579	-15.881561	-91.850494
3	6.484507	17.314789	-5.754576	-21.826047	56.531934
4	6.827589	41.110603	5.709650	-36.868356	30.400690

In [7]:
```python
# Drop outcome + omics
exclude = ['ID', 'risk_score', 'weighted_composite_risk'] + omics_cols
X_exposure_all = master_df.drop(columns=exclude, errors='ignore')

# Keep only numeric columns
X_exposure_numeric = X_exposure_all.select_dtypes(include=[np.number]).copy()
X_exposure_numeric.head()
```

Out[7]:

	h_abs_ratio_preg_Log	h_no2_ratio_preg_Log	h_pm10_ratio_preg_None	h_pm25_ratio_preg_None	hs_no2_dy_hs_h_Log	hs_no2_wk_hs_h_Log	hs_no2_yr_hs_h_Log	hs_pm10_dy_hs_h_None	hs_pm10_wk_hs_h_None	hs_pm10_yr_hs_h_None	...	methylome_child_risk_y	methyl
0	0.896711	2.872304	25.948498	17.433798	2.530279	2.583284	2.612098	22.535828	20.850005	31.399067	...	3.701852	
1	0.892538	2.980008	25.897739	18.470850	1.928600	2.652479	2.761064	14.077763	29.141274	31.250535	...	-0.700855	
2	0.778723	3.056501	26.087347	18.711547	2.882591	2.591756	2.356163	46.859096	31.530981	27.516001	...	-7.840449	
3	0.089056	3.089157	14.991380	16.409771	1.390750	2.456717	2.403247	29.817442	25.232778	23.965263	...	-0.679772	
4	0.604781	3.848211	35.197296	14.889958	3.204449	3.499594	3.307663	29.817442	24.891465	24.754238	...	0.059042	

5 rows × 206 columns

In [20]:
```python
scaler = StandardScaler()
X_exposure_scaled = pd.DataFrame(
    scaler.fit_transform(X_exposure_numeric),
    columns=X_exposure_numeric.columns,
    index=X_exposure_numeric.index
)


# Sanity check
print("X_exposure shape:", X_exposure_scaled.shape)
```

```
X_exposure shape: (1301, 206)
```

CV Training Function

In [14]:
```python
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score

def cross_validate_model(X, y, model, model_name, n_splits=5):
    skf = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

    aucs, accs, f1s = [], [], []

    for train_idx, val_idx in skf.split(X, y):
        X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
        y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

        model.fit(X_train, y_train)
        y_pred = model.predict(X_val)
        y_prob = model.predict_proba(X_val)[:, 1]

        aucs.append(roc_auc_score(y_val, y_prob))
        accs.append(accuracy_score(y_val, y_pred))
        f1s.append(f1_score(y_val, y_pred))

    return {
        'Model': model_name,
        'AUC': f"{np.mean(aucs):.3f} ± {np.std(aucs):.3f}",
        'Accuracy': f"{np.mean(accs):.3f} ± {np.std(accs):.3f}",
        'F1 Score': f"{np.mean(f1s):.3f} ± {np.std(f1s):.3f}"
    }
```

In [15]:
```python
# Logistic Regression (LASSO-style)
logreg = LogisticRegression(penalty='l1', solver='saga', max_iter=5000, random_state=42)

# XGBoost Classifier
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', max_depth=3, learning_rate=0.1, n_estimators=100, random_state=42)

# Random Forest
rf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
```

In [21]:
```python
# Store results here
results = []
# Logistic Regression (LASSO-style)
logreg = LogisticRegression(
    penalty='l1',
    solver='saga',
    max_iter=5000,
    random_state=42
)


# Exposure-only
results.append(cross_validate_model(X_exposure_scaled, y_bin, logreg, "LogReg - Exposure"))
results.append(cross_validate_model(X_exposure_scaled, y_bin, xgb, "XGBoost - Exposure"))
results.append(cross_validate_model(X_exposure_scaled, y_bin, rf, "RandomForest - Exposure"))
```

In [22]:
```python
import pandas as pd
results_df = pd.DataFrame(results)
results_df = results_df[['Model', 'AUC', 'Accuracy', 'F1 Score']]  # ordered

import IPython.display as disp
disp.display(results_df)
```

	Model	AUC	Accuracy	F1 Score
0	LogReg - Exposure	0.997 ± 0.001	0.968 ± 0.011	0.936 ± 0.021
1	XGBoost - Exposure	0.986 ± 0.007	0.940 ± 0.007	0.872 ± 0.015
2	RandomForest - Exposure	0.961 ± 0.013	0.875 ± 0.025	0.671 ± 0.090

LASSO with Cross-Validation and λ Reporting

In [18]:
```python
from sklearn.linear_model import LogisticRegressionCV
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Run LogisticRegressionCV with L1 penalty
lasso_cv = LogisticRegressionCV(
    Cs=10,                # Try 10 values of inverse λ
    cv=5,                 # 5-fold internal CV
    penalty='l1',
    solver='saga',
    scoring='roc_auc',
    max_iter=5000,
    random_state=42,
    n_jobs=-1
)

# Scale exposure data and fit
pipeline = make_pipeline(StandardScaler(), lasso_cv)
pipeline.fit(X_exposure_numeric, y_bin)

# Best inverse regularization parameter (C), λ = 1/C
best_C = lasso_cv.C_[0]
lambda_val = 1 / best_C
print(f"🔍 Best C (inverse λ): {best_C:.4f}")
print(f"🔧 Selected λ: {lambda_val:.4f}")

# Coefficients
coef = lasso_cv.coef_.flatten()
features = X_exposure_numeric.columns

# Top non-zero predictors
nonzero_mask = coef != 0
top_features = list(zip(features[nonzero_mask], coef[nonzero_mask]))

print(f"\n📈 Non-zero predictors retained: {len(top_features)}")
for name, weight in sorted(top_features, key=lambda x: abs(x[1]), reverse=True):
    print(f"{name:50s} {weight:.4f}")
```

```
🔍 Best C (inverse λ): 0.3594
🔧 Selected λ: 2.7826

📈 Non-zero predictors retained: 65
total_metabo_risk_x                               8.1906
total_transcript_risk_x                           4.8564
total_protein_risk_x                              2.7154
methylome_child_risk_x                            1.3594
methylome_child_risk_y                            1.3594
h_clf_preg_Log                                    0.4836
metabo_child_risk_x                               0.3954
metabo_child_risk_y                               0.3954
hs_pfna_c_Log2                                    -0.3125
protein_child_risk_x                              0.3100
protein_child_risk_y                              0.3100
h_Benzene_Log                                     -0.3003
h_PM_Log                                          -0.2861
hs_hum_mt_hs_h_None                               -0.2444
hs_popdens_s_Sqrt                                 -0.2273
hs_uvdvf_mt_hs_h_None                             0.2119
hs_cd_m_Log2                                      0.2020
hs_prpa_madj_Log2                                 -0.2015
hs_bpa_madj_Log2                                  -0.2005
hs_meohp_cadj_Log2                                -0.1967
h_thm_preg_Log                                    0.1881
hs_dmp_madj_Log2                                  0.1658
hs_pfoa_m_Log2                                    -0.1648
hs_pfna_m_Log2                                    -0.1556
hs_ndvi100_s_None                                 0.1384
h_folic_t1_None                                   0.1314
h_TEX_Log                                         -0.1217
hs_greenyn300_s_None                              -0.0999
hs_pm25abs_dy_hs_h_Log                            -0.0941
hs_trcs_madj_Log2                                 0.0912
hs_pm10_yr_hs_h_None                              0.0902
hs_no2_dy_hs_h_Log                                -0.0882
hs_pcb118_madj_Log2                               -0.0867
hs_pfoa_c_Log2                                    0.0777
hs_fdensity300_h_Log                              0.0693
hs_cd_c_Log2                                      -0.0678
hs_dep_madj_Log2                                  -0.0677
h_bro_preg_Log                                    0.0636
hs_uvdvf_wk_hs_h_None                             0.0609
hs_pbde47_madj_Log2                               0.0591
hs_etpa_madj_Log2                                 -0.0531
hs_pb_m_Log2                                      0.0498
hs_mvpa_prd_alt_None                              0.0488
hs_pbde153_madj_Log2                              -0.0487
hs_co_m_Log2                                      0.0469
hs_mepa_madj_Log2                                 -0.0409
hs_cs_m_Log2                                      0.0399
mirna_child_risk_x                                0.0281
mirna_child_risk_y                                0.0281
hs_pcb170_cadj_Log2                               0.0266
hs_pb_c_Log2                                      0.0247
hs_trafload_h_pow1over3                           -0.0237
hs_etpa_cadj_Log2                                 0.0236
e3_alcpreg_yn_None                                0.0189
h_Absorbance_Log                                  -0.0143
hs_mbzp_madj_Log2                                 -0.0141
h_frichness300_preg_None                          0.0138
hs_hg_m_Log2                                      -0.0130
h_walkability_mean_preg_None                      0.0121
hs_dmp_cadj_Log2                                  0.0085
transcript_pregnancy_risk_x                       0.0084
transcript_pregnancy_risk_y                       0.0084
hs_pet_cat_r2_None                                0.0059
hs_dep_cadj_Log2                                  -0.0032
hs_dmtp_cadj_Log2                                 -0.0031
```

In [ ]:
```python
# Build DataFrame for LASSO plot
import pandas as pd

df_lasso = pd.DataFrame(top_features, columns=["Feature", "LASSO Coef"])
df_lasso["|LASSO Coef|"] = df_lasso["LASSO Coef"].abs()
df_lasso = df_lasso.sort_values(by="|LASSO Coef|", ascending=False).reset_index(drop=True)
```

In [ ]:
```python
import matplotlib.pyplot as plt

# Plot top 20 features by absolute LASSO coefficient
top_n = 20
df_plot = df_lasso.head(top_n).sort_values(by='|LASSO Coef|', ascending=True)

plt.figure(figsize=(10, 8))
plt.barh(df_plot['Feature'], df_plot['|LASSO Coef|'], color='steelblue')
plt.xlabel('Absolute Coefficient')
plt.title(f'Top {top_n} Features Retained by LASSO (CV)')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```