NORMAL 90/10 Validation

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score, roc_curve
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
exposure_df = pd.read_csv("../hello_extracted.csv")
methylome = pd.read_csv("methylome_risk_scores.csv")
transcriptome = pd.read_csv("transcript_risk_scores.csv")
mirna = pd.read_csv("mirna_risk_scores.csv")
proteome = pd.read_csv("protein_risk_scores.csv")
metabolome = pd.read_csv("metabolite_risk_scores.csv")
```

In [3]:
```
train_expo, test_expo = train_test_split(
    exposure_df,
    test_size=0.10,
    stratify=exposure_df["FAS_cat_None"],  # pick a stratification column (e.g. FAS or site)
    random_state=42
)
```

In [5]:
```
def merge_omics(df):
    for f in [methylome, transcriptome, mirna, proteome, metabolome]:
        df = df.merge(f, on="ID", how="left")
    return df

train_df = merge_omics(train_expo.copy())
test_df = merge_omics(test_expo.copy())
```

In [6]:
```
# Correct omics column names and weights
train_df["final_risk"] = (
    0.18 * train_df["total_methylome_risk"] +
    0.12 * train_df["total_transcript_risk"] +
    0.20 * train_df["total_mirna_risk"] +
    0.25 * train_df["total_protein_risk"] +
    0.25 * train_df["total_metabo_risk"]
)

test_df["final_risk"] = (
    0.18 * test_df["total_methylome_risk"] +
    0.12 * test_df["total_transcript_risk"] +
    0.20 * test_df["total_mirna_risk"] +
    0.25 * test_df["total_protein_risk"] +
    0.25 * test_df["total_metabo_risk"]
)
```

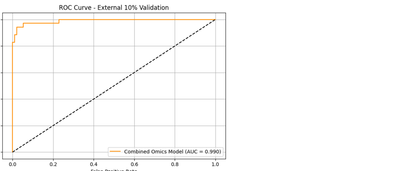
In [7]:
```
threshold = np.percentile(train_df["final_risk"], 75)

train_df["label"] = (train_df["final_risk"] >= threshold).astype(int)
test_df["label"] = (test_df["final_risk"] >= threshold).astype(int)

print(f"Risk Threshold (Train 75th percentile): {threshold:.3f}")
print(f"Train Positives: {train_df['label'].sum()} / {len(train_df)}")
print(f"Test Positives: {test_df['label'].sum()} / {len(test_df)}")
```

```
Risk Threshold (Train 75th percentile): 9.648
Train Positives: 259 / 1170
Test Positives: 35 / 131
```

In [11]:
```
omics_cols = [
    'total_methylome_risk',
    'total_transcript_risk',
    'total_mirna_risk',
    'total_protein_risk',
    'total_metabo_risk'
]

X_train = train_df[omics_cols]
y_train = train_df["label"]

X_test = test_df[omics_cols]
y_test = test_df["label"]

clf = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
clf.fit(X_train, y_train)

y_prob = clf.predict_proba(X_test)[:, 1]
y_pred = clf.predict(X_test)

auc_val = roc_auc_score(y_test, y_prob)
acc_val = accuracy_score(y_test, y_pred)
f1_val = f1_score(y_test, y_pred)

print(f"[True 90/10 Validation] AUC: {auc_val:.3f} | Accuracy: {acc_val:.3f} | F1 Score: {f1_val:.3f}")
```

```
[True 90/10 Validation] AUC: 0.990 | Accuracy: 0.962 | F1 Score: 0.928
```

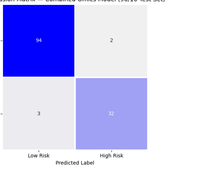
In [13]:
```
fpr, tpr, _ = roc_curve(y_test, y_prob)

plt.figure(figsize=(5,5))
plt.plot(fpr, tpr, label=f'Combined Omics Model (AUC = {auc_val:.3f})', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - External 10% Validation")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [17]:
```
from sklearn.metrics import classification_report, confusion_matrix, precision_score, recall_score

# Binary metrics
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_prob)
acc = roc_auc_score(y_test, y_prob)
cm = confusion_matrix(y_test, y_pred)

print("🔍 Final Classification Metrics (10% Test Set)")
print(f"AUC:          {auc:.3f}")
print(f"Accuracy:     {acc:.3f}")
print(f"F1 Score:     {f1:.3f}")
print(f"Precision:    {precision:.3f}")
print(f"Recall:       {recall:.3f}")
print("\nConfusion Matrix:")
print(cm)
```

```
🔍 Final Classification Metrics (10% Test Set)
AUC:          0.990
Accuracy:     0.962
F1 Score:     0.928
Precision:    0.941
Recall:       0.914

Confusion Matrix:
[[94  2]
 [ 3 32]]
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Generate confusion matrix again if needed
cm = confusion_matrix(y_test, y_pred)
labels = ["Low Risk", "High Risk"]

# Plot heatmap
plt.figure(figsize=(6,5))
sns.heatmap(cm,
            annot=True,
            fmt="d",
            cmap=sns.light_palette("blue", reverse=False, as_cmap=True),
            xticklabels=labels,
            yticklabels=labels,
            cbar=False,
            linewidths=1.5,
            linecolor='white',
            square=True)

plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix — Combined Omics Model (90/10 Test Set)")
plt.tight_layout()
plt.show()
```



Stratified K-Fold Cross-Validation

In [19]:
```
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.metrics import (
    roc_auc_score, accuracy_score, f1_score,
    precision_score, recall_score, roc_curve, confusion_matrix
)
from xgboost import XGBClassifier

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

In [20]:
```
# Load exposure and risk score datasets
exposure_df = pd.read_csv("../hello_extracted.csv")
methylome = pd.read_csv("methylome_risk_scores.csv")
transcriptome = pd.read_csv("transcript_risk_scores.csv")
mirna = pd.read_csv("mirna_risk_scores.csv")
proteome = pd.read_csv("protein_risk_scores.csv")
metabolome = pd.read_csv("metabolite_risk_scores.csv")
```

In [21]:
```
from sklearn.model_selection import train_test_split

train_expo, test_expo = train_test_split(
    exposure_df,
    test_size=0.10,
    stratify=exposure_df["FAS_cat_None"],  # stratify by SES
    random_state=42
)
```

In [23]:
```
def merge_omics(df):
    for f in [methylome, transcriptome, mirna, proteome, metabolome]:
        df = df.merge(f, on="ID", how="left")
    return df

train_df = merge_omics(train_expo.copy())
test_df = merge_omics(test_expo.copy())
```

In [24]:
```
# Weighted risk calculation
weights = {
    "total_methylome_risk": 0.18,
    "total_transcript_risk": 0.12,
    "total_mirna_risk": 0.20,
    "total_protein_risk": 0.25,
    "total_metabo_risk": 0.25,
}

def compute_weighted_risk(df):
    return sum(w * df[col] for col, w in weights.items())

train_df["final_risk"] = compute_weighted_risk(train_df)
test_df["final_risk"] = compute_weighted_risk(test_df)

# Threshold for binarization
threshold = np.percentile(train_df["final_risk"], 75)
train_df["label"] = (train_df["final_risk"] >= threshold).astype(int)
test_df["label"] = (test_df["final_risk"] >= threshold).astype(int)

print(f"Threshold (75th percentile): {threshold:.3f}")
print(f"Train Positives: {train_df['label'].sum()} / {len(train_df)}")
print(f"Test Positives: {test_df['label'].sum()} / {len(test_df)}")
```

```
Threshold (75th percentile): 9.648
Train Positives: 259 / 1170
Test Positives: 35 / 131
```

In [25]:
```
from tqdm.auto import tqdm

# Features and labels
X = train_df[list(weights.keys())].values
y = train_df["label"].values

# 20-fold stratified CV
skf = StratifiedKFold(n_splits=20, shuffle=True, random_state=42)

# Metrics storage
auc_scores, acc_scores, f1_scores, prec_scores, rec_scores = [], [], [], [], []
mean_fpr = np.linspace(0, 1, 100)
tpr_list = []

# Cross-validation loop with tqdm
for fold, (train_idx, val_idx) in enumerate(tqdm(skf.split(X, y), total=20), 1):
    X_train, X_val = X[train_idx], X[val_idx]
    y_train, y_val = y[train_idx], y[val_idx]

    model = XGBClassifier(use_label_encoder=False, eval_metric="logloss", random_state=fold)
    model.fit(X_train, y_train)

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

    auc = roc_auc_score(y_val, y_prob)
    acc = accuracy_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)
    prec = precision_score(y_val, y_pred)
    rec = recall_score(y_val, y_pred)

    auc_scores.append(auc)
    acc_scores.append(acc)
    f1_scores.append(f1)
    prec_scores.append(prec)
    rec_scores.append(rec)

    fpr, tpr, _ = roc_curve(y_val, y_prob)
    tpr_interp = np.interp(mean_fpr, fpr, tpr)
    tpr_interp[0] = 0.0
    tpr_list.append(tpr_interp)
```

```
100%|██████████| 20/20 [00:01<00:00, 18.12it/s]
```

In [26]:
```
# Summary statistics for 20-fold CV
def summarize_metric(name, scores):
    print(f"{name:<10}: {np.mean(scores):.3f} ± {np.std(scores):.3f}")

print("🔁 20-Fold Cross-Validation Metrics (Combined Omics Model):\n")
summarize_metric("AUC", auc_scores)
summarize_metric("Accuracy", acc_scores)
summarize_metric("F1 Score", f1_scores)
summarize_metric("Precision", prec_scores)
summarize_metric("Recall", rec_scores)
```

```
🔁 20-Fold Cross-Validation Metrics (Combined Omics Model):

AUC       : 0.994 ± 0.007
Accuracy  : 0.962 ± 0.011
F1 Score  : 0.921 ± 0.044
Precision : 0.947 ± 0.056
Recall    : 0.901 ± 0.065
```

In [29]:
```
# Mean and std of TPR across folds
mean_tpr = np.mean(tpr_list, axis=0)
std_tpr = np.std(tpr_list, axis=0)
mean_tpr[-1] = 1.0

plt.figure(figsize=(7,7))
plt.plot(mean_fpr, mean_tpr, label=f"Mean ROC (AUC = {np.mean(auc_scores):.3f})", color='darkblue')
plt.fill_between(mean_fpr,
                 np.clip(mean_tpr - std_tpr, 0, 1),
                 np.clip(mean_tpr + std_tpr, 0, 1),
                 color='lightblue', alpha=0.4, label="±1 SD")

plt.plot([0, 1], [0, 1], 'k--', linewidth=1)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Mean ROC Curve — 20-Fold Cross-Validation (Combined Omics)")
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```