SHAP Explainability for PAM50 Subtype Prediction

Author: Julian Borges, M.D. — HMS GCSRT 2025
Capstone: *"Hidden Biases in AI-Powered Genomic Subtyping of Breast Cancer"*

Table of Contents

1. Install Dependencies

```
# Install Required Packages (Colab-safe)
!pip install shap xgboost pandas matplotlib seaborn
```

```
Requirement already satisfied: shap in /usr/local/lib/python3.11/dist-packages (0.47.1)
Requirement already satisfied: xgboost in /usr/local/lib/python3.11/dist-packages (2.1.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from shap) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from shap) (1.14.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from shap) (1.6.1)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.11/dist-packages (from shap) (4.67.1)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.11/dist-packages (from shap) (24.2)
Requirement already satisfied: slicer==0.0.8 in /usr/local/lib/python3.11/dist-packages (from shap) (0.0.8)
Requirement already satisfied: numba>=0.54 in /usr/local/lib/python3.11/dist-packages (from shap) (0.60.0)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.11/dist-packages (from shap) (3.1.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/dist-packages (from shap) (4.13.1)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.11/dist-packages (from xgboost) (2.21.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.54->shap) (0.4
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->shap) (3.6.0)
```

2. Upload and Load Dataset

```
from google.colab import files
uploaded = files.upload()

import pandas as pd
import io

# Get the actual filename from the upload dictionary
filename = list(uploaded.keys())[0]

# Load the CSV using the actual uploaded filename
data = pd.read_csv(io.BytesIO(uploaded[filename]))
data.head()
```

Choose Files AI-BIAS cle…dataset.csv

- **AI-BIAS cleaned_TCGA_PAM50_model_dataset.csv**(text/csv) - 18545 bytes, last modified: 4/17/2025 - 100% done

```
Saving AI-BIAS cleaned_TCGA_PAM50_model_dataset.csv to AI-BIAS cleaned_TCGA_PAM50_model_dataset (1).csv
```

	pam50_numeric	er_binary	pr_binary	her2_binary	age_at_diagnosis	stage_simplified
0	4	1	0	0	70.0	1
1	2	1	1	0	59.0	2
2	2	1	1	0	56.0	1
3	2	1	1	0	54.0	2
4	2	1	1	0	61.0	2

Próximas etapas: Gerar código com data | Ver gráficos recomendados | New interactive sheet

```
# Split into features and target
X = data.drop(columns=['pam50_numeric'])
y = data['pam50_numeric']
```

3. Model Training

```
# Split into features and target
X = data.drop(columns=['pam50_numeric'])
y = data['pam50_numeric']  # No need to shift

from xgboost import XGBClassifier

model = XGBClassifier(
    objective='multi:softprob',
    num_class=5,
    eval_metric='mlogloss',
    n_jobs=-1
)
model.fit(X, y)
```

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=None,
              n_jobs=-1, num_class=5, num_parallel_tree=None, ...)
```

```
# Split correctly
X = data.drop(columns=['pam50_numeric'])
y = data['pam50_numeric']
```

```
model = XGBClassifier(
    objective='multi:softprob',
    num_class=5,
    eval_metric='mlogloss',
    n_jobs=-1
)

model.fit(X, y)  # ✅ y should be in range [0, 1, 2, 3, 4]
```

```
XGBClassifier

XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='mlogloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=None,
              n_jobs=-1, num_class=5, num_parallel_tree=None, ...)
```

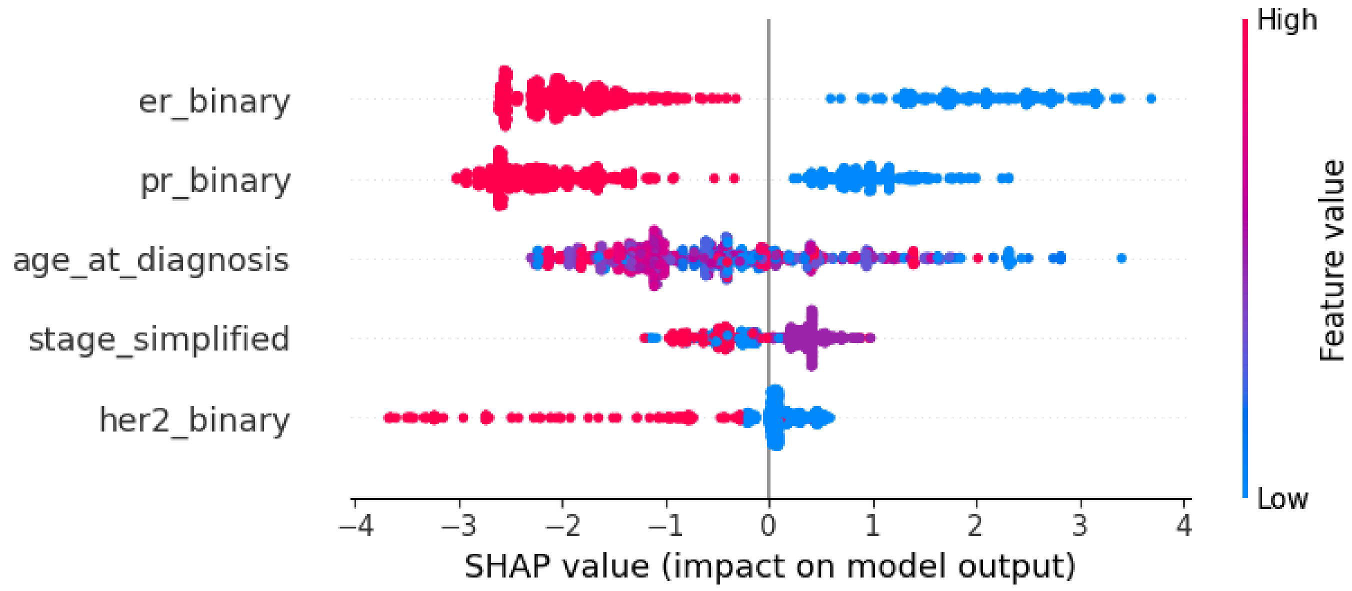
4. SHAP Global & Subtype Explainability

```
# Compute SHAP values
import shap
explainer = shap.Explainer(model)
shap_values = explainer(X)
```

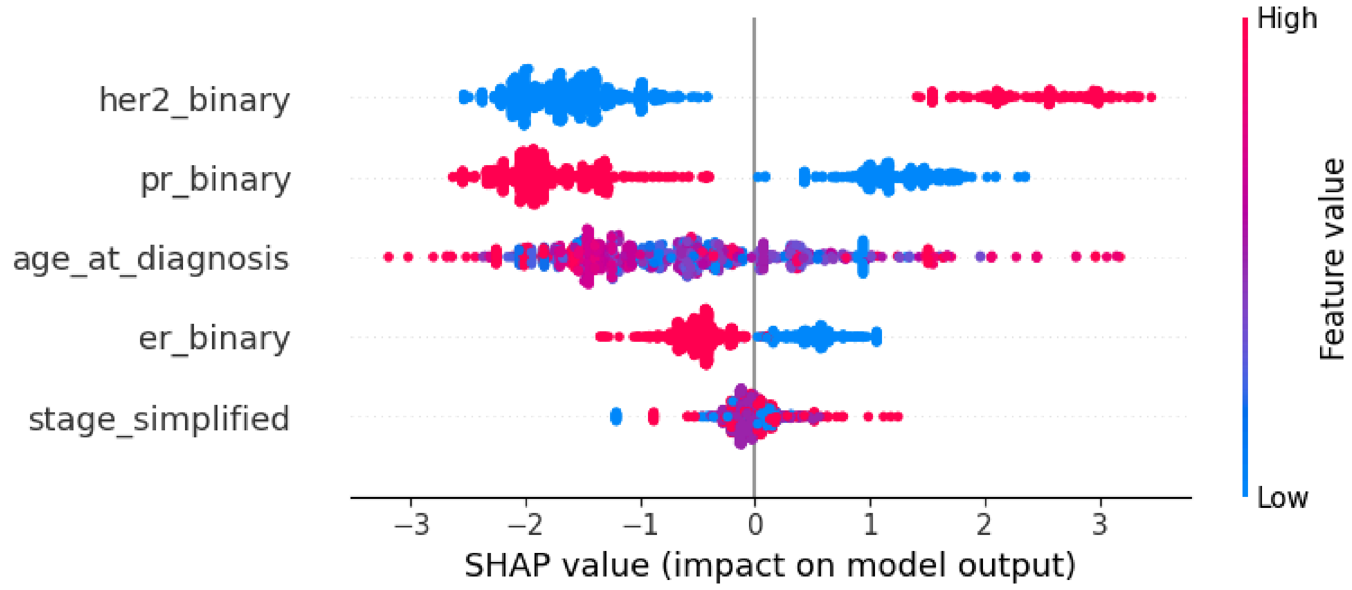
```
import numpy as np

# Check for non-zero values in the SHAP array
print("SHAP value shape:", shap_values.values.shape)
print("SHAP value stats (abs mean):", np.abs(shap_values.values).mean())
```

```
SHAP value shape: (1231, 5, 5)
SHAP value stats (abs mean): 0.7441172
```

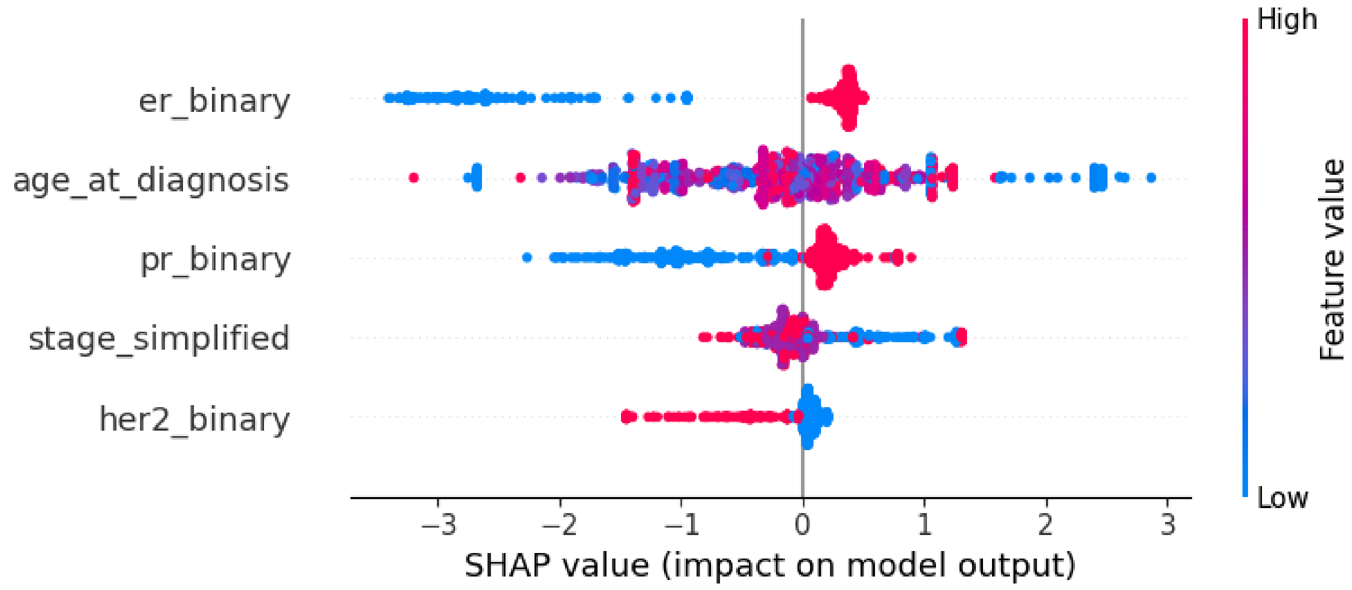
```
for i in range(5):
    shap.summary_plot(shap_values[:, :, i], X, show=True, class_names=[f"Class {i}"])
```



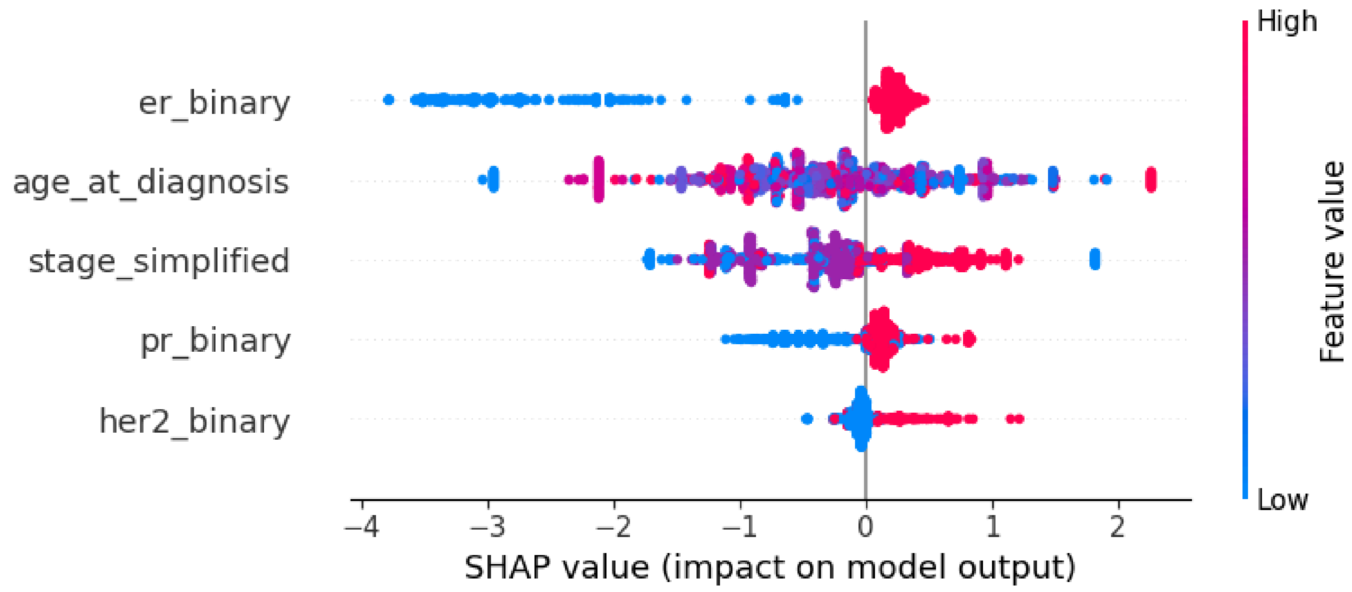
er_binary
pr_binary
age_at_diagnosis
stage_simplified
her2_binary
−4 −3 −2 −1 0 1 2 3 4
SHAP value (impact on model output)
High
Feature value
Low



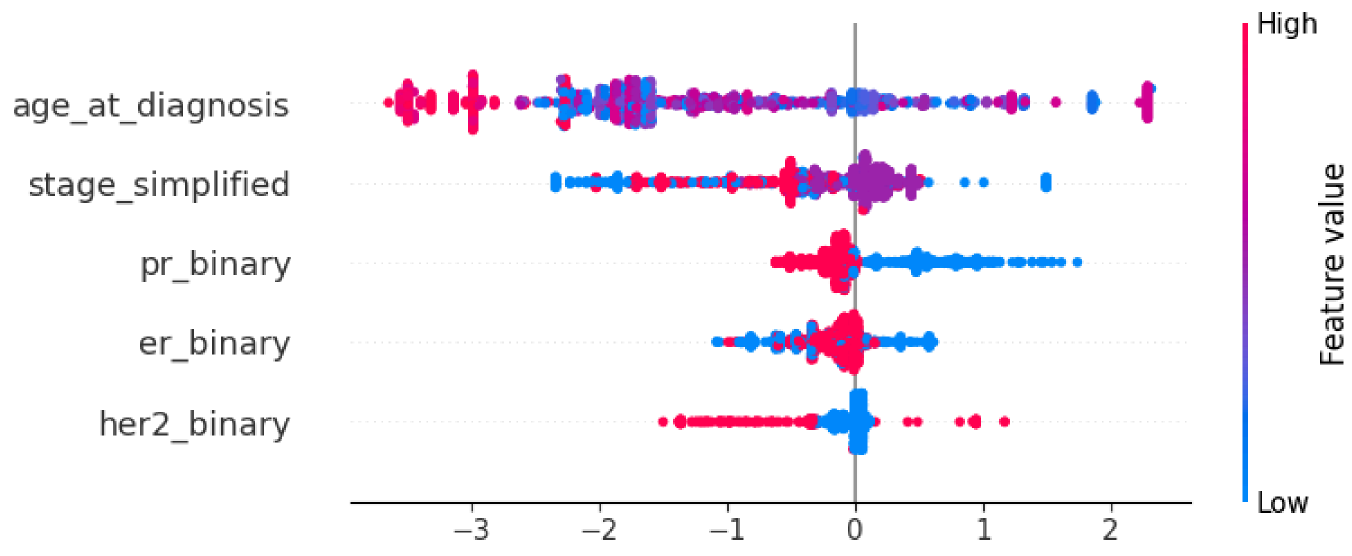
her2_binary
pr_binary
age_at_diagnosis
er_binary
stage_simplified
−3 −2 −1 0 1 2 3
SHAP value (impact on model output)
High
Feature value
Low



er_binary
age_at_diagnosis
pr_binary
stage_simplified
her2_binary
−3 −2 −1 0 1 2 3
SHAP value (impact on model output)
High
Feature value
Low



er_binary
age_at_diagnosis
stage_simplified
pr_binary
her2_binary
−4 −3 −2 −1 0 1 2
SHAP value (impact on model output)
High
Feature value
Low



age_at_diagnosis
stage_simplified
pr_binary
er_binary
her2_binary
−3 −2 −1 0 1 2
High
Feature value
Low

SHAP value (impact on model output)

Interpretation of This SHAP Plot:

The SHAP values are extremely narrow and centered around 0

All features (e.g., her2_binary, pr_binary) have low impact magnitude

There's no clear separation by feature value, even by color

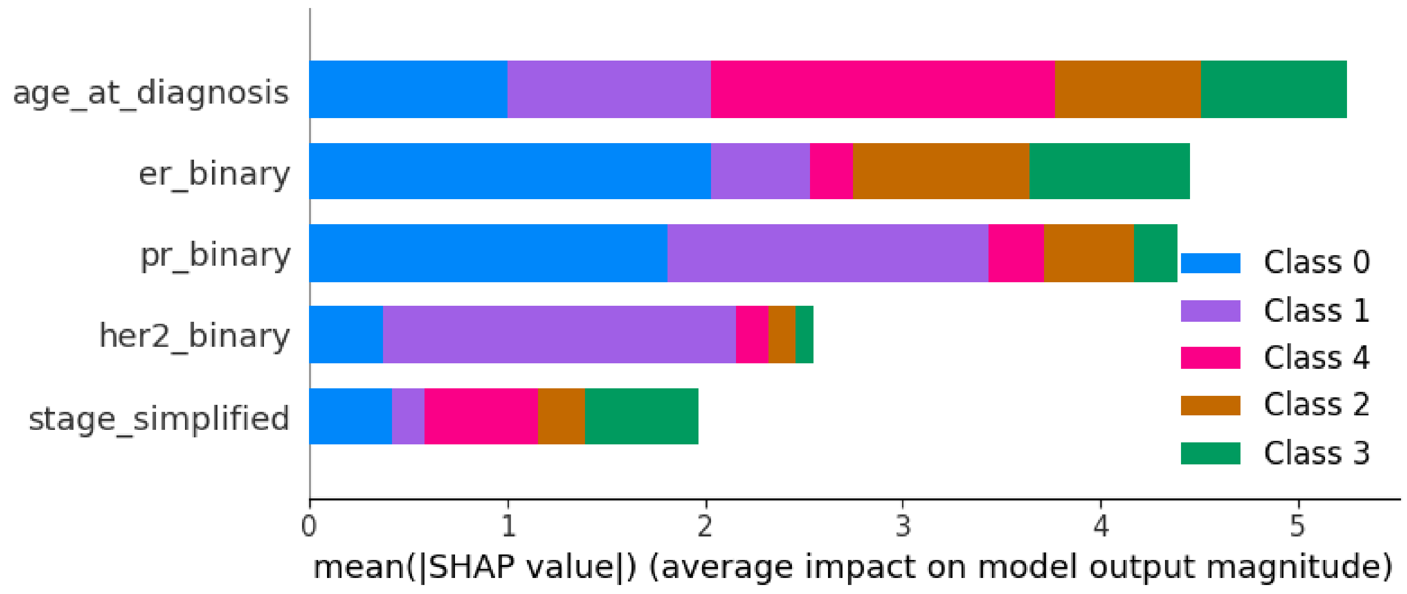
That suggests this current model is:

Not well-calibrated

Possibly underfitting or

Trained on low-signal or too-small data

```
# Visualize SHAP summary
shap.summary_plot(shap_values, X)
```




```
# Optional: Save SHAP values to download later
import numpy as np
np.save('shap_values.npy', shap_values.values)
```

```
# Subtype-wise SHAP Bar Plots
import matplotlib.pyplot as plt

label_map = {
    0: "Basal",
    1: "Her2-enriched",
    2: "Luminal A",
    3: "Luminal B",
    4: "Normal-like"
}

for i in range(5):
    shap.plots.bar(shap_values[:, :, i], max_display=10, show=False)
    plt.title(f"SHAP Feature Importance for {label_map[i]}")
    plt.savefig(f"SHAP_Barplot_{label_map[i].replace(' ', '_')}.png", bbox_inches="tight", dpi=300)
    plt.show()
```