WQF7009 - Alternative Assessment 2024

In this alternative assessment, we are required to predict heart disease and build an explainable AI model that explains the features that contribute to the prediction.

We are data scientists, and given training and testing datasets to build the models from.

Task 1

In this task, we load the data and perform basic exploratory data analysis, data preprocessing and dataset splits. Below are the relevant code sections taken from the Colab notebook.

Dataset information

Our training dataset, training_set_heart.csv, contains the following features. The testing dataset, testing_set_heart.csv contains the same features, except the target variable.

Features	Type	Description
Age	Numerical	Age of the patient in years
Sex	Categorical	Male/Female
CP	Categorical	chest pain type ([typical angina = 0, atypical angina = 1, non-anginal = 2, asymptomatic = 3]
Trestbps	Numerical	resting blood pressure
Chol	Numerical	serum cholesterol in mg/dl
Fbs	Categorical	if fasting blood sugar > 120 mg/dl [Yes = 1, No = 0]
Restecg	Categorical	resting electrocardiographic results [normal = 0, stt abnormality = 1, lv hypertrophy = 2]
Thalach	Numerical	maximum heart rate achieved
Exang	Categorical	exercise-induced angina [True = 1, False = 0]
Oldpeak	Numerical	ST depression induced by exercise relative to rest
Slope	Categorical	the slope of the peak exercise ST segment [0, 1, 2]
Ca	Numerical	number of major vessels (0-3) colored by fluoroscopy
Thal	Categorical	0, 1, 2, 3
Target	Numerical	The predicted attribute [Heart disease = 1, No heart disease = 0]

For this task, we perform the following data preprocessing steps:

1. **Import libraries and load the data:** We import and load the libraries as below. Code snippets are taken from the Colab notebook for this assessment.

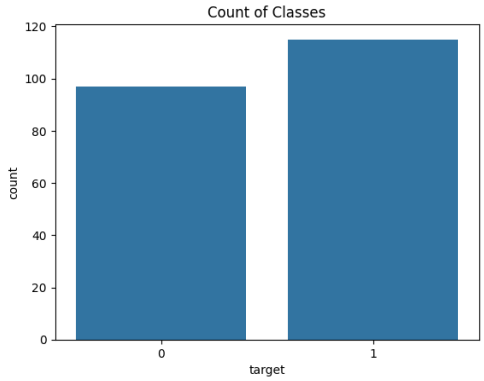
```
[ ] #Import libraries

    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns

    from sklearn.model_selection import train_test_split, GridSearchCV
    from sklearn.metrics import confusion_matrix, classification_report
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score
    from sklearn.preprocessing import StandardScaler, OneHotEncoder

    import shap
```

```
[ ] df = pd.read_csv('/content/training.csv')
    test = pd.read_csv('/content/testing.csv')
```

---

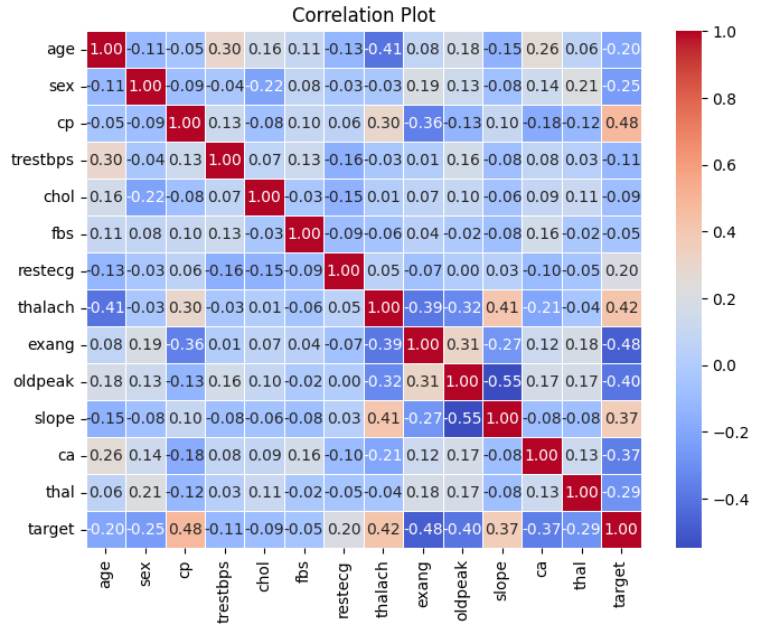
2. **Perform basic EDA:** We perform basic exploratory data analysis on the dataset. While our codebook provides a more comprehensive analysis, we look at the only the key findings.

   a. Count of classes: Overall, the dataset is slightly imbalanced. However, as this imbalance is minimal, we choose not to oversample the imbalanced class.

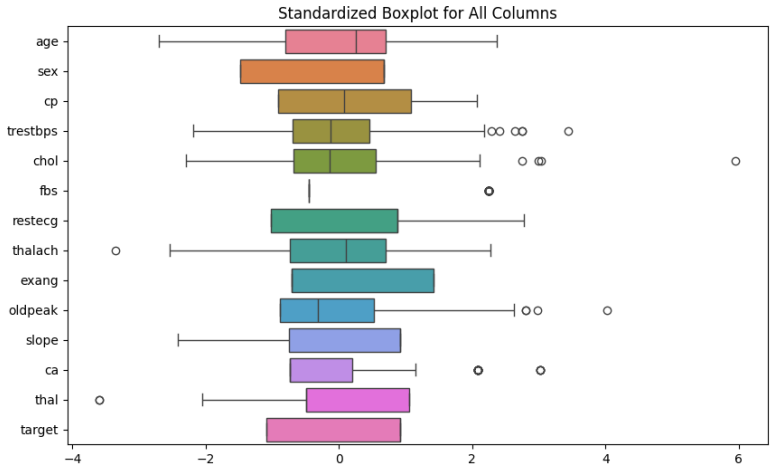


   b. Pairplots: We produce pairplots for all the features in the dataset.

c. Correlation plot: We produce a correlation plot for all the features in the dataset. Overall, oldpeak and slope have the lowest correlation of -0.55, while cp and target have the highest non-diagonal correlation of 0.48.



d. Standardised boxplots for all columns: As the data ranges vary widely across columns, we standardize the data and plot boxplots for each feature. Overall, trestbps, chol, fbs, oldpeak and ca have multiple outliers for their respective plots, while thalach and thal have 1 each. The sex and target columns do not have any whiskers, as they are binary values of 0 and 1. The fbs column has the least spread.



3. **Encode the data:** As this dataset contains many categorical variables, we choose to use the OneHotEncoder function for these variables. The ‘sex’, ‘cp’, ‘fbs’, ‘restecg’, ‘exang’, ‘slope’ and ‘thal’ variables are encoded.

```
[ ]  from sklearn.preprocessing import OneHotEncoder


     # From the dataset description, the following are categorical variables

     cat_col = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'thal']

     # Perform one-hot encoding
     df_encoded = pd.get_dummies(df, columns=cat_col)


     # Display the resulting DataFrame
     print(df_encoded)
```

4. **Separate data into input and target variables:** We separate the data into input and target variables to model the data later on.

```
[ ]  # Separate dataset into target and features

     X = df_encoded.drop('target', axis = 1)
     y = df_encoded['target']
```

5. **Scale data:** For this dataset, we choose to scale the data as the different numerical values in the dataset are on different measurement scales. Not scaling the data could lead to adverse results when fitting the model.

```
[ ]  # Scale data as they are all measured on different metrics

     scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

6. **Split data into training and testing sets:** As our testing set does not contain target variables, we split the training dataset into training and validation sets. We employ an 80/20 split for our training and validation sets.

```
[ ]  #Split dataset into training and testing datasets

     X_train, X_valid, y_train, y_valid = train_test_split(X_scaled, y, test_size = 0.2, random_state = 261)
```

Task 2

For this dataset, we choose to build a random forest classifier to predict heart disease. We took the following steps:

1. **Initialise the model**: We initialised the random forest classifier and set the seed at random_state = 261 for reproducibility purposes.
2. **Conduct hyperparameter tuning**: We used the GridSearchCV function to select the best hyperparameters. We also tune GridSearchCV to optimise the hyperparameters to achieve the highest F1-scores. The hyperparameters we are tuning are 'n_estimators', 'max_depth', 'min_samples_split' and 'min_samples_leaf'.
3. **Fit the best model**: The best hyperparameters chosen by GridSearchCV are then used to fit the training data, X_train and y_train, on the random forest model. The best parameters are 'n_estimators' = 50, 'max_depth' = None, 'min_samples_split' = 2 and 'min_samples_leaf' = 2.
4. **Make predictions on the validation set**: We then make predictions on X_valid, the validation dataset using the best estimator found by GridSearchCV. We call these values y_pred.
5. **Evaluate the model**: We then calculate the F1-scores for y_valid, the real target variables, and y_pred, our predicted values. With the best estimators, we achieve an **F1-score of 0.8182.** The high F1-score indicates that there is a good balance between precision and recall, which implies that the model is effective at correctly identifying both positive and negative heart disease cases.

Code snippet

```
[ ] #### NOW FOR SCORING F1-SCORE
```

```
# Define the hyperparameter grid to search
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create GridSearchCV
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='f1')

# Fit the model to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters and best estimator from the grid search
best_params = grid_search.best_params_
best_estimator = grid_search.best_estimator_

# Make predictions on the test set using the best estimator
y_pred = best_estimator.predict(X_valid)
```

```
[ ] # Evaluate the model
f1 = f1_score(y_valid, y_pred)
print(f"Best Parameters: {best_params}")
print(f"F1-score with Best Estimator: {f1:.4f}")
```

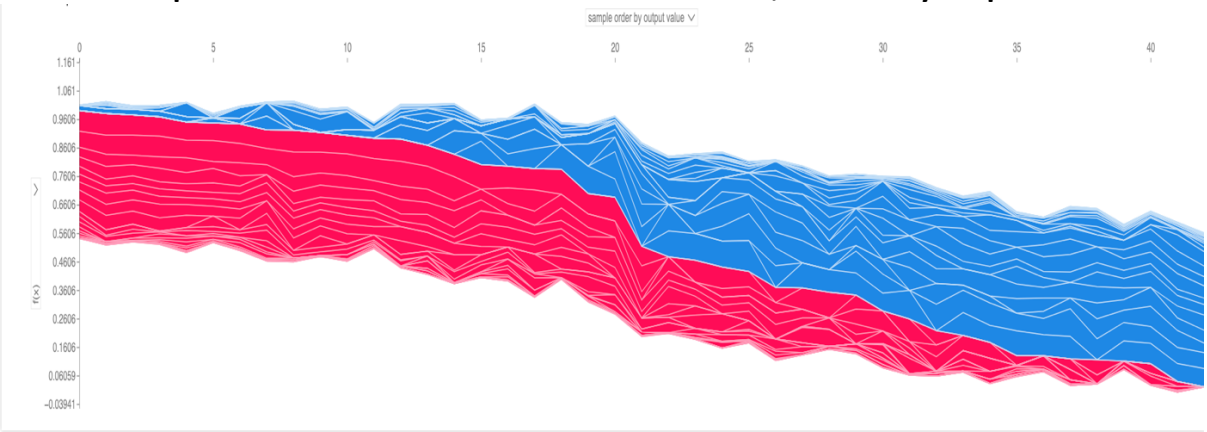
```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 50}
F1-score with Best Estimator: 0.8182
```

Task 3

In this task, we use a SHAP explainer to derive SHAP values for the machine learning model, from the generated plot write our inferences. Below are the steps taken:

1. **Initiliase our model**: Using the best hyperparameters, we initialise our random forest classifier as our model.
2. **Create SHAP explainer**: We use shap.TreeExplainer on our model to calculate SHAP values. We use TreeExplainer specifically for its unique capabilities to calculate SHAP values for tree based models such as Random Forest classifiers.
3. **Calculate SHAP values for the dataset**: We calculate SHAP values for the entire validation dataset. The values are stored as shap_values
4. **Calculate f(x)**: f(x) represents the threshold for the random forest model to predict heart disease as 1. F(x) is 0.5606.
5. **Visualise plots**: We then generate a force plot and summary plot for the SHAP values.

Force plot of SHAP values on the validation dataset, ordered by output value



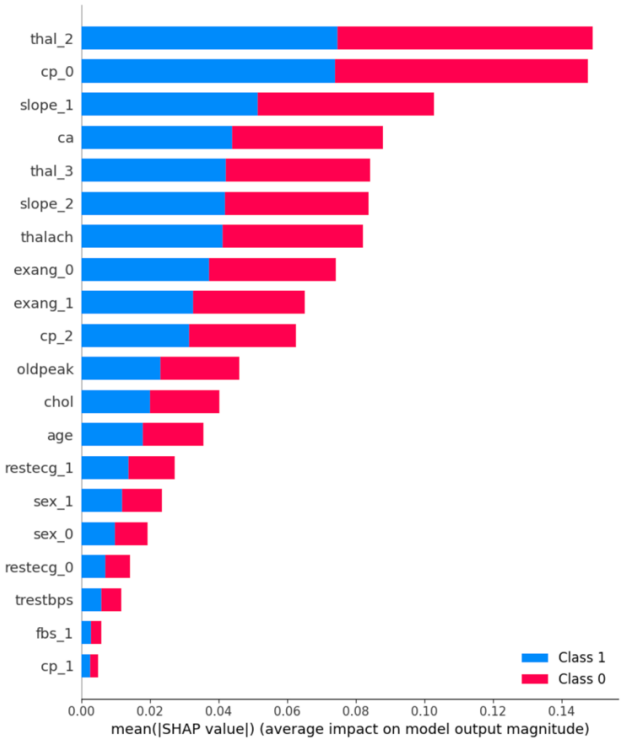
Code snippet

```
[115] model = grid_search.best_estimator_
```

```
# %% Create SHAP explainer
explainer = shap.TreeExplainer(model)

# Calculate shapley values for test data
start_index = 0
end_index = len(X_valid) + 1
shap_values = explainer.shap_values(X_valid[start_index:end_index])
X_valid[start_index:end_index]
```

```
[36] # %% >> Visualize global predictions for the model
shap.initjs()
# Force plot
prediction = model.predict(X_valid[start_index:end_index])[0]
print(f"The RF predicted: {prediction}")
shap.force_plot(explainer.expected_value[1],
                shap_values[1],
                X_valid[start_index:end_index], feature_names = feature_names) # for values
```

Summary plot of the SHAP values on the validation dataset



Code snippet

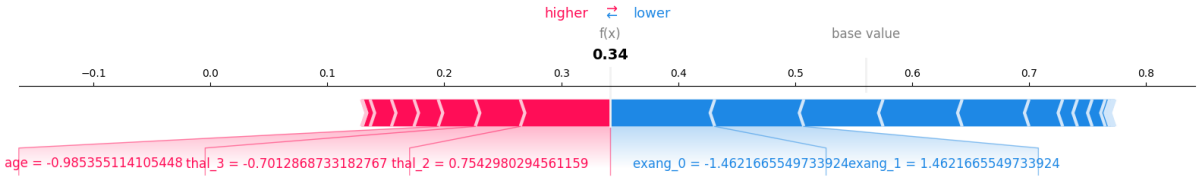
```
# %% >> Visualize global features
# Feature summary
shap.summary_plot(shap_values, X_valid, feature_names = feature_names)
```

From the two plots, we can observe:

- Based on the summary plot, thal_2, cp_0 and slope_1 are the most important factors contribution to f(x), the model prediction for both Class 0 and Class 1. Thal_2 and cp_0 make up roughly 15% of the contribution to model predictions, while slope_1 represents roughly 10% of feature contribution. Thal_2 represents reversible defect, while cp_0 represents typical angina, and slope_1 references the first peak of exercise ST segment.
- From the summary plot, trestbps, fbs_1 and cp_1 contribute the least to the model's prediction decision for both Class 0 and Class 1. These represent the resting blood pressure, fasting blood sugar and atypical angina.
- From the force plot of the SHAP values for class 1 of the entire validation dataset, roughly 20 patients are predicted to have heart disease, as their f(x) values exceed the threshold of f(x) = 0.5606.

Task 4

SHAP Force Plot on the first row of the test dataset



Code snippet

```
[127] # %% Create SHAP explainer
      explainer1 = shap.TreeExplainer(model)
      # Calculate shapley values for test data
      start_index1 = 0
      end_index1 = start_index1 + 1
```

```
shap_values1 = explainer1.shap_values(X_test[start_index1:end_index1])
print(shap_values1[0].shape)
```

```
(1, 26)
```

+ Code + Text

```
[136] explainer1.expected_value[1]
```

```
0.5605917159763313
```

```
[130] feature_names = test_encoded.columns
```

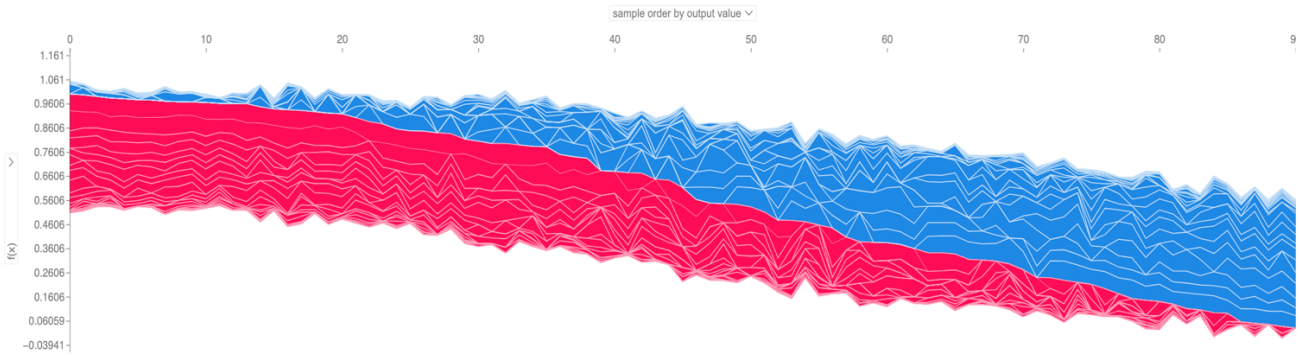
```
[137] # %% >> Visualize global predictions for the model
      shap.initjs()
      # Force plot
      prediction = model.predict(X_test[start_index1:end_index1])[0]
      print(f"The RF predicted: {prediction}")
      shap.force_plot(explainer1.expected_value[1],
                      shap_values1[1],
                      X_test[start_index1:end_index1], matplotlib = True, feature_names = feature_names) # for values
```

From the force plot on the first row of the test dataset, we can observe:

- The expected value of this instance is f(x) = 0.34, which indicates that the random forest model predicted 0 for this patient, and they don't have heart disease.
- Exang_0, which indicates there was no exercised-induced angina, contributed the most to the model prediction.
- Meanwhile, thal_2 and exang_1 round up the top 3 feature contributions towards the model prediction. Thal_2 represents reversible defect.

Task 5

SHAP Force plot for all the rows of the test dataset



Code snippet

```
[47] # %% Create SHAP explainer
     explainer2 = shap.TreeExplainer(model)
     # Calculate shapley values for test data
     start_index2 = 0
     end_index2 = len(X_test) + 1
```

```
     shap_values2 = explainer2.shap_values(X_test[start_index2:end_index2])
     print(shap_values2[0].shape)
```

```
     (91, 26)
```

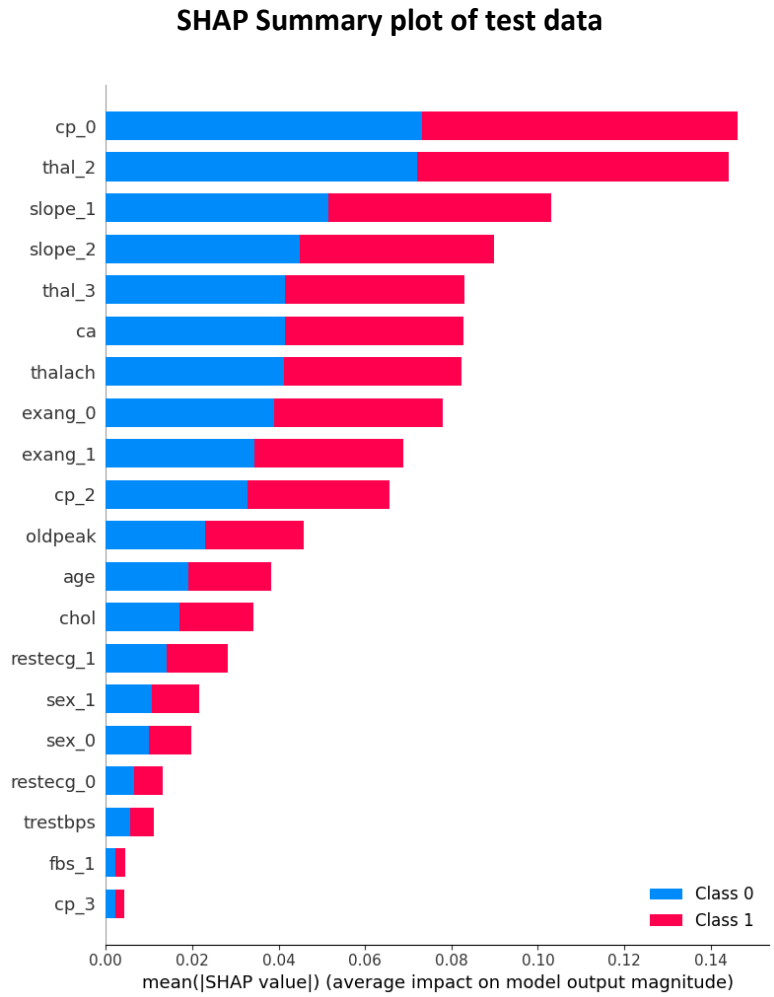
```
[49] print(explainer2.expected_value[1])

     0.5605917159763313
```

```
[50] # %% >> Visualize global predictions for the model
     shap.initjs()
     # Force plot
     prediction = model.predict(X_test[start_index2:end_index2])[0]
     print(f"The RF predicted: {prediction}")
     shap.force_plot(explainer2.expected_value[1],
                     shap_values2[1],
                     X_test[start_index2:end_index2], feature_names = feature_names) # for values
```

From the SHAP force plot for all rows of the test dataset, we can observe:

- Roughly half of the observations lie above the threshold of f(x) = 0.5606, indicating that these observations were classified as target=1, indicating they have heart disease.

Task 6



Code snippet

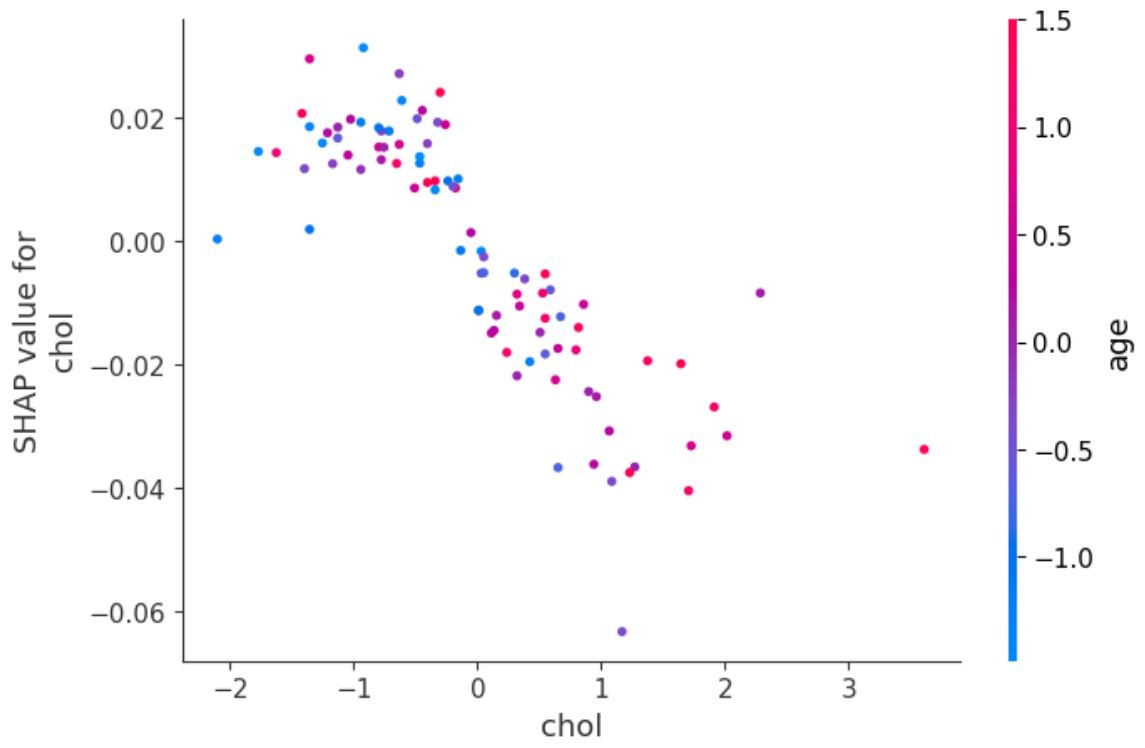
```
# %% >> Visualize global features
# Feature summary
shap.summary_plot(shap_values2, X_test, feature_names = feature_names)
```

From the SHAP summary plot of the test data, we can observe:

- Cp_0, thal_2 and slope_1 are the top 3 features that contribute to the model prediction, f(x). These features make up roughly 40% of the model explanation.
- Cp_0 and thal_2 contribute around the same weightage of 15% each of the model explanation.
- Meanwhile, fbs_1 and cp_3 contribute the least towards model explanation at less than 1% feature contribution each.
- The summary plot for the test data and validation data have the same top 3 features that contribute to model predictions, but differ for the features that contribute the least.
- While this summary plot explains feature weightage for the entire dataset, the top feature contributions for each instance of data may differ. Comparing this summary plot to the force plot in task 4, the force plot places more weightage on the exang_0, thal_2 and exang_1 features.

Task 7

SHAP dependence plot for chol feature, with age interaction index



Code snippet

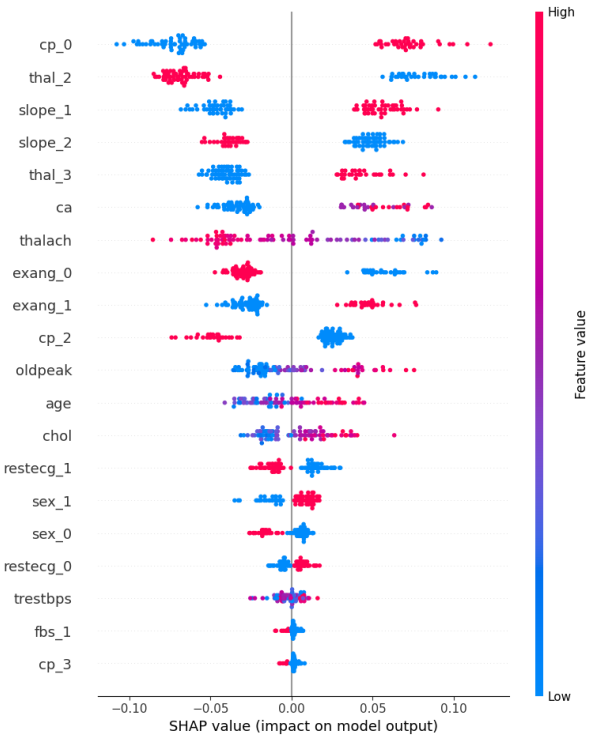
```
shap.dependence_plot(2, shap_values2[1], X_test, feature_names = feature_names, interaction_index = 0)
```

From the SHAP dependence plot for the chol feature, we can observe:

- There is an inverse relationship between chol values and the SHAP values for the chol feature. As chol values increase, the SHAP values for chol decrease.
- This suggests that for patients with lower cholesterol values, the model “push” the prediction higher towards the target of 1.
- The random spread of the age interaction index indicates that there is no clear age interaction effect with cholesterol, indicating a weak correlation between the two features.

Task 8

Beeswarm summary for Class 1 (right)



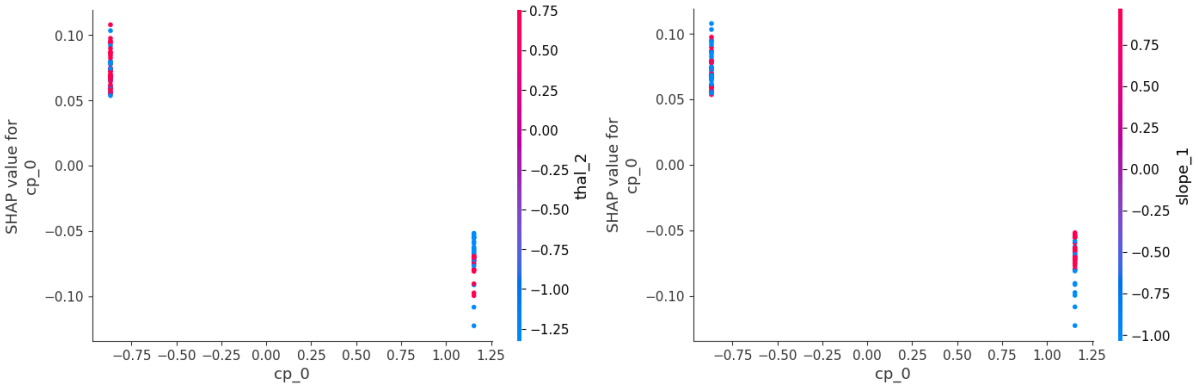
Code snippet

```
shap.summary_plot(shap_values2[1], X_test, feature_names = feature_names)
```

From the beeswarm summary plot, we can observe:

- Typical angina chest pain (cp_0 = 1) have positive SHAP values, with prediction leaning towards heart disease.
- Reversible defects (thal_2 = 1) have negative SHAP values, with predictions leaning towards no heart disease.
- Features such as thalach, oldpeak, age, chol and trestbps have complex interactions, leading to colours being mixed.

Dependence plots for cp_0, with thal_2 (left) and slope_1 (right) interaction



Code snippet

```
# dependence plot for cp_0, with interaction effects slope 1
shap.dependence_plot(8, shap_values2[1], X_test, feature_names = feature_names, interaction_index = 20)
```

```
# dependence plot cp_0, with thal_2 interaction effects
shap.dependence_plot(8, shap_values2[1], X_test, feature_names = feature_names, interaction_index = 24)
```

As cp_0, thal_2 and slope_1 are the features that explain the most on model prediction, we can infer from the above graphs:

- Generally, SHAP values decrease for those who have typical angina chest pain (cp_0 = 1), while they increase for those without typical anginal chest pain.
- Those with typical anginal chest pain generally do not have a reversible defect (thal_2 = 0). Conversely, those with different chest pains (cp_1, cp_2, cp_3) have reversible defects (thal_2 = 1)
- Meanwhile, those with typical angina chest pain (cp_0 = 1) have slope_1 (slope 1 of the peak exercise ST segment)

Link to Colab file:

https://colab.research.google.com/drive/1FlN6ISJ7UiPdcG9GY7VhrP4axxnR_RjY?usp=sharing