INTRODUCTION

Hello! My name is Jessica Kim!

I'm glad you'll be the wonderful individual who will take this project to the next level. I was an intern for the summer of 2022 who had the amazing opportunity to work under Dr. Urbanowicz and work with his research, STREAMLINE. I'm a Health Administration major with a minor in Computer Science at CSUN (California State Unversity, Northridge). I started this internship with beginner coding experience (especially with Python) and no experience with Machine Learning. This notebook, as a bit basic and not the most efficient it can be, is something I am proud to have been able to produce (more like a personal breakthrough) so I really hope what I wrote is clear enough to understand how I went about it. Either way, I would love to see the progress as you fix it because it would help me better understand how I can code more efficiently.

OVERVIEW

This document is the most recent version of the bugs and tasks that still need to be done for the SHAP Jupyter Notebook. The notebook uses the latest SHAP package version, version 0.41.0. When opening the notebook, you will find:

- A summary of the purpose and functions of the notebook itself
- Goals of the SHAP code
- A progress log of bugs/issues that I've encountered (considering beginner coding & ML experience) and have fixed based on the structure of the code
- Updates on what other tasks need to be done
- A possible future notebook change idea that may or may not work entirely with the code

Much of the code at the beginning where parameters, datasets, and other necessary variables are loaded/unpickled is from the STREAMLINE code. This was done to keep code consistent but also to help me move things along and work on the actual computation for SHAP values & figures.

This is just a summarization of issues I've had with the SHAP package and implementing it with STREAMLINE but I'd be more than happy to talk one-on-one about it. It would probably be easier to talk over the phone than explain it in words haha.

Please refer to the presentation to find the hierarchy of how the SHAP values and figures are saved when you run this notebook.

CONCERNS W/ SHAP NOTEBOOK CODE

Hopefully, you've had the chance to look through the section called "Progress of Updates/Fixes" and as you can see, it's just a mix of issues/bugs I've had to fix either because I didn't know how to do it or it wasn't done correctly.

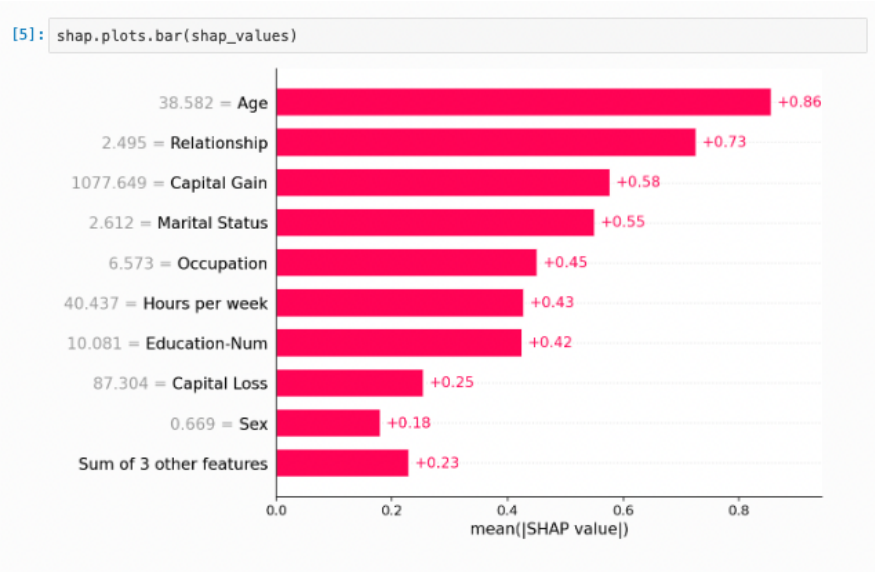
The issues I did have with implementing SHAP from a post-run of STREAMLINE were with implementing some function calls such as the .shap_values() call to generate the SHAP values or shap.bar_plot() for figures. I'm not entirely sure if it's because I don't have a full understanding in general, the changes in implementation methods of SHAP, a mix of both, or if STREAMLINE is setting up things a bit different than what would normally be used to create SHAP values & figures. Here is a breakdown of the issues/concerns:

- Understanding/figuring out how to create SHAP Explainers properly
  - **Usual explainer format**
    - **explainer(model, train_dataset_X)**
    - Train dataset X is used as a parameter because it serves as "background data" so that the explainer has reference when it comes to computing SHAP values on Test dataset X
  - .Explainer() → format depending on the model
    - Gaussian Naive Bayes
      - The usual format is explainer(model, train_dataset_X) BUT for Naive Bayes only allows for **shap.Explainer(model.predict, train_dataset_X)** → even "model.predict_proba" doesn't work
  - .TreeExplainer()
    - **shap.TreeExplainer(model)**
    - Only needs to use the trained model as the parameter
    - Doesn't require a background dataset
  - .LinearExplainer()
    - Logistic Regression & Linear Regression
    - **shap.LinearExplainer(model, train_dataset_X)**

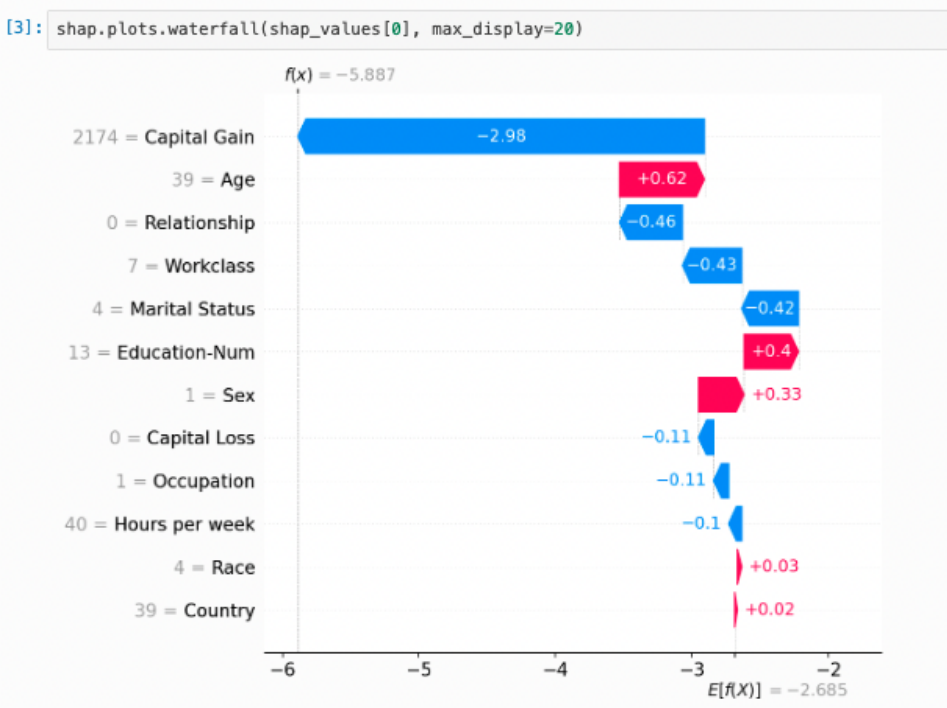
- Understanding how to calculate SHAP values for different ML models
  - Gaussian Naive Bayes
    - can't use .**shap_values()** whereas other models such as Logistic Regression or any tree models can use it

- Linear models & Tree models
  - CAN use **.shap_values()**
  - Additional parameters for tree-models can are used to account for the nature & behavior of tree-model predictions

- Function calls for certain SHAP figures don't work for certain models
  - Trust me…I've created a dummy notebook for the model I was looking into, such as Decision Trees, to try to create SHAP values and figures with the function calls I've been having issues using for the trained models from STREAMLINE
  - I've left some comments within the cells for the SHAP summary figures that are supposed to work for the models
    - **Bar plot**
      - A figure that shows SHAP value in bar plot figure as opposed to just feature ranking names in a bar plot form → not necessarily working
      - The most updated fix for generating this model but haven't been able to make it work for the models (as seen in the issues section on GitHub) is **shap.plots._bar.bar_legacy()**



    - **Waterfall plot**
      - A figure that shows SHAP values of each feature that has a positive and negative influence on the final model predictions (w/values labeled)
      - Tried to make this work for Naive Bayes, Logistic Regression, and Extreme Gradient Boosting
      - The most updated fix for generating this plot is **shap.plots._waterfall.waterfall_legacy()**



- Because these models are mainly **BINARY CLASSIFICATION**, some plot function calls require the parameter ".expected_value" or ".expected_value[0]" or ".expected_value[1]" → this issue being that there's a gap in consistent implementation between Naive Bayes, linear models, and tree models
  - The 0 (zero) or 1 refers to the Classes of the predictions that the models generated
  - This format applies similarly to the parameter for shap_values
  - **Example:** Looking at Class 0 of the model predictions
    - shap.decision_plot(expected_value[0], shap_values[0], feature_names=feature_names, show=False)

- Force Plots are an optional run for this notebook so it is user-specified but saving the figures is an issue for this
  - When run, you'll notice that the figures **can only be saved for Naive Bayes**
  - Other models can only have the figures displayed in the output cell when running the notebook → haven't been able to figure out why

- Overall efficiency and structure of the notebook code need a definite improvement
- The notebook has not been tested on other trained models from STREAMLINE such as KNN, SVM, and ANN

A few links to look at:

- **SHAP Explainers:** https://shap.readthedocs.io/en/latest/api.html#explainers
- **SHAP Plot Examples:** https://shap.readthedocs.io/en/latest/api_examples.html#plots
- **Explaining a model that uses standardized features (a look at a linear model):** https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/linear_models/Explaining%20a%20model%20that%20uses%20standardized%20features.html
- **SHAP GitHub:** https://github.com/slundberg/shap
- **Cohort Bar Plot Error issue:** https://github.com/slundberg/shap/issues/2406
- **Understanding Shapley Values:** https://www.kaggle.com/code/iamleonie/understanding-shapley-values/notebook
- **Using SHAP to Explain Machine Learning Models:** https://blog.kthais.com/using-shap-to-explain-machine-learning-models-3f8f9c3b1f5e
- **Explain your model predictions with Shapley Values:** https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-shapley-values/notebook
- **SHAP Values:** https://www.kaggle.com/code/dansbecker/shap-values
- **Shap Values feature selection for Medical Fraud:** https://www.kaggle.com/code/dan7cor/shap-values-feature-selection-for-medical-fraud