MLops Pipeline for Bias Detection and Mitigation in Heart Disease Diagnosis Models

Team name: Git Commit

MLOPS PROBLEM STATEMENT:

Our goal is to enhance fairness, reliability, and equitable predictions across diverse demographic groups. We'll emphasize the integration of ensemble techniques to improve model accuracy.

Approach:

Data Preprocessing and Exploration:

- **Normalize Data**: Pre-processed the dataset by handling missing values, outliers, and class imbalances.
- **Stratified Sampling**: Ensured diverse representation by stratifying data based on demographic groups.

Bias Detection:

Gender analysis is an important aspect of bias detection and mitigation in machine learning. Helped us understand how our Heart Disease diagnosis models perform across different gender groups and identify any potential disparities or discrimination.

Accuracy of Naive Bayes model for men: 82.35294117647058

Accuracy of Naive Bayes model for women: 91.30434782608695

These accuracy scores indicate that our Naive Bayes model **performs better on women than on men**. This could be a **sign of gender bias** in your model, data, or evaluation metrics. To mitigate this bias, we used:

- Applied debiasing technique - **Reweighting**
- To mitigate bias in our models, we applied **AIF3600**, a comprehensive open-source library for bias detection and mitigation.

- AIF3600 was our choice of tool for bias mitigation, as it provides a wide range of metrics, explanations, and algorithms for datasets and models.

```
from aif360.algorithms.preprocessing import Reweighing
from aif360.datasets import BinaryLabelDataset

# Assuming you've already loaded your dataset into 'df'
privileged_group = {'sex': 1}  # Define the privileged group based on your dataset
unprivileged_group = {'sex': 0}  # Define the unprivileged group based on your dataset
privileged_groups = [privileged_group]
unprivileged_groups = [unprivileged_group]

# reweighing = Reweighing(unprivileged_groups=[{"sex": 0}], privileged_groups=[{"sex": 1}])
# reweighing.fit(df)
# dataset_transformed = reweighing.transform(df)
```

```
train_bld = BinaryLabelDataset(favorable_label=1, unfavorable_label=0, df=pd.concat([X_train, y_train], axis=1), label_names=['t
test_bld = BinaryLabelDataset(favorable_label=1, unfavorable_label=0, df=pd.concat([X_test, y_test], axis=1), label_names=['targ

# Train and apply reweighing algorithm to mitigate bias
rw = Reweighing(unprivileged_groups=unprivileged_groups, privileged_groups=privileged_groups)
train_bld_reweighed = rw.fit_transform(train_bld)
```

```
# Train a new model on the reweighed dataset
reweighed_model = RandomForestClassifier(n_estimators=100, random_state=42)
reweighed_model.fit(train_bld_reweighed.features, train_bld_reweighed.labels.ravel())
```

```
RandomForestClassifier(random_state=42)
```

```
# # Evaluate the reweighed model
y_pred_reweighed = reweighed_model.predict(test_bld.features)
reweighed_accuracy = accuracy_score(test_bld.labels, y_pred_reweighed)
print(f"Reweighed Model Accuracy: {reweighed_accuracy}")
```

```
Reweighed Model Accuracy: 0.9873417721518988
```

Ensemble Techniques for Model Optimization:

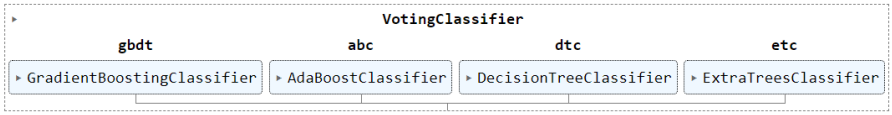
We have combined four different models, used **Voting classifiers, using hard voting strategy,** each of which have its own strengths and weaknesses, and aggregated their predictions to obtain a final prediction. Our **Ensemble model achieved an accuracy of 0.987**, which indicates that your ensemble technique was effective.

Voting classifiers are a type of **ensemble technique** that **combine the predictions of multiple base models and select the majority vote as the final prediction.** They can improve the accuracy and robustness of your machine learning models by leveraging the diversity and strengths of different models.

```
[83]: voting = VotingClassifier(estimators=[('gbdt', gbdt), ('abc', abc), ('dtc', dtc), ('etc', etc)],voting='soft')

[84]: voting.fit(X_train, y_train)
```

[84]:




```
[85]: y_pred = voting.predict(X_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(precision_score(y_test, y_pred))

disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, y_pred))
disp.plot()
plt.show()

# voting model is most accurate and precise
```

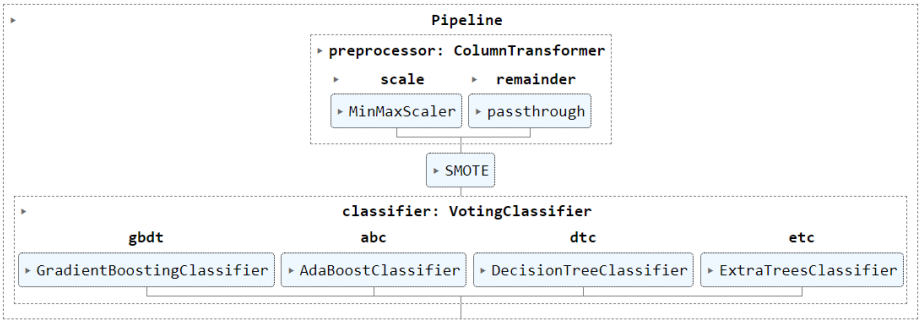
```
0.9873417721518988
[[22  0]
 [ 1 56]]
```

Model Used	Accuracy
Logistic Regression	86.34146341463415
Decision Tree	85.36585365853658
Random Forest	100.0 (Overfitting)
Naïve Bayes	Accuracy of Naive Bayes model for men: 82.35294117647058 Accuracy of Naive Bayes model for women: 91.30434782608695
Extreme Gradient Boost	95.1219512195122
K-Neighbours Classifier	87.8048780487805

Pipeline:

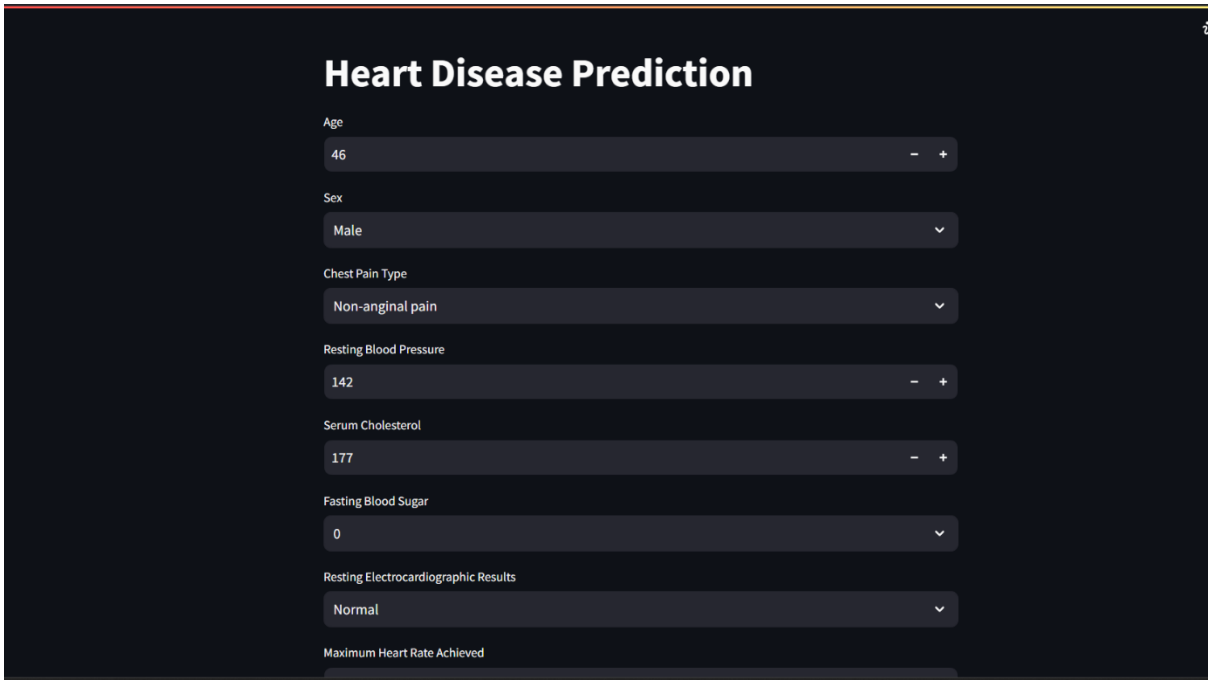
```
In [92]: # Fit the pipeline to the training data
         pipeline.fit(X_train, y_train)
```

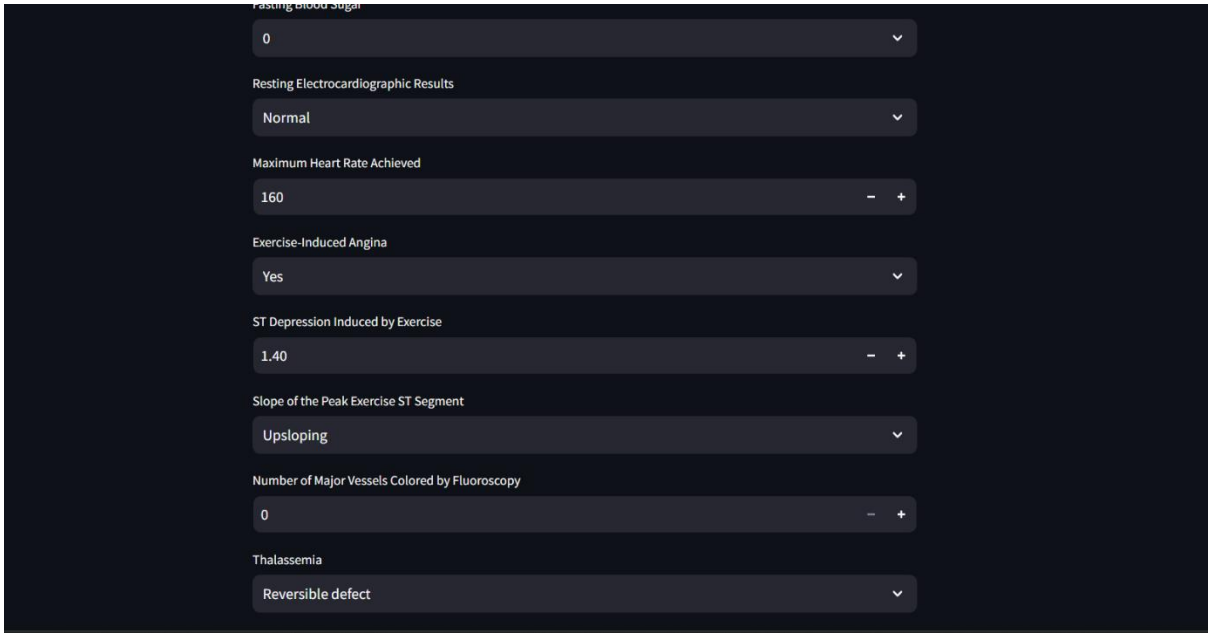
Out[92]:



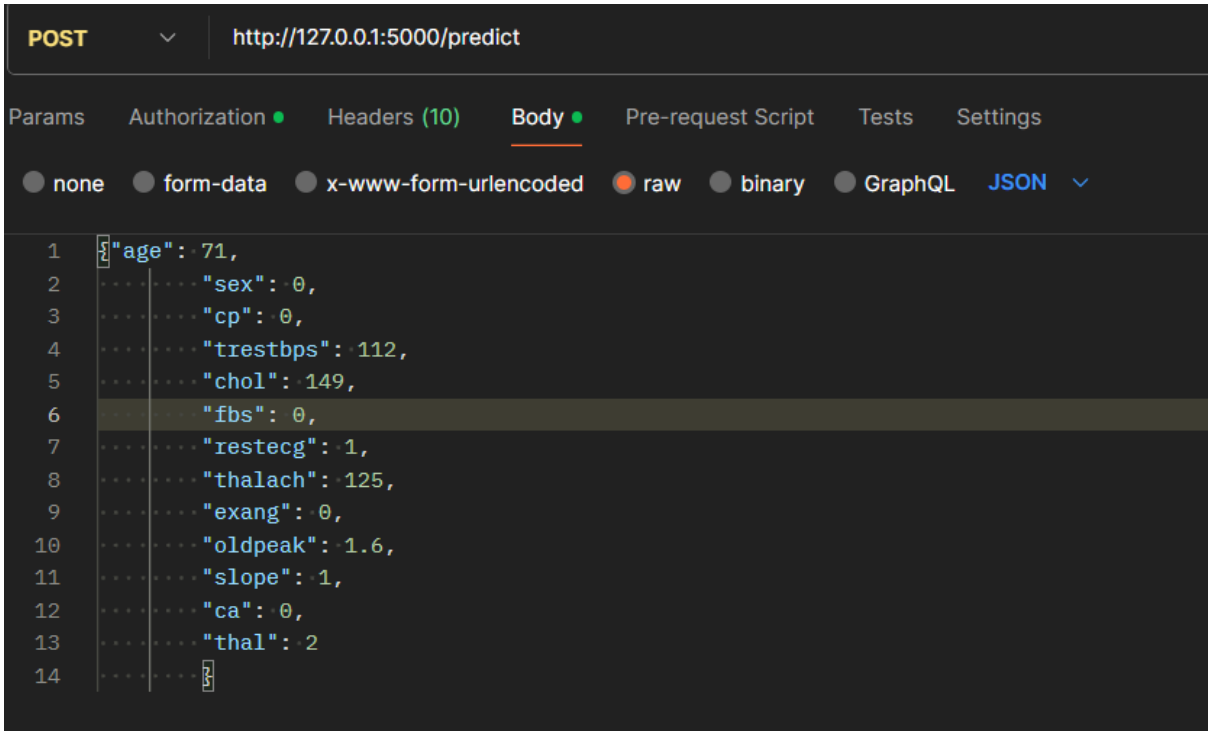
User Interface:

We have used Streamlit for making the user interface, as it is easy and scalable Python UI Framework.





Deployment:



Github Repo: https://github.com/RacHiT101/MLOps-Heart_Disesase_Pred/tree/main