Introduction:

For this exercise I experimented with different preprocessing methods and their effect on various ML models as well as with and without hyperparameter optimization. I went with numerical preprocessors because every categorical preprocessor I tried had extremely long runtimes that I would just cancel after 15+ minutes of running.

Dataset Description:

I used the wine quality dataset[1] for this and picked the white wine dataset because it has more entries than the red dataset. This data includes 11 features and a target value called "quality" that is between 0 and 10.

[1]: https://archive-beta.ics.uci.edu/dataset/186/wine+quality

Experimental Setup:

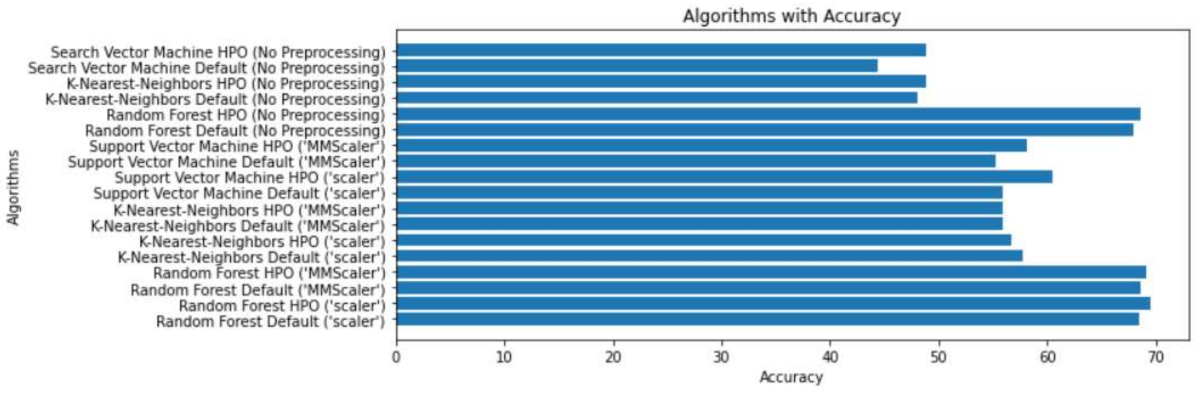
For these experiments I picked a few models to test on, Random Forest because it has been the most accurate in every test on the wine dataset, K Nearest Neighbors because I wanted to how much preprocessing would improve it, and Support Vector Machine because I wanted a third model, and it was used in all the examples on scikit-learn's documentation.

Then I picked a couple preprocessing methods, I decided to use two different scaler methods because I thought those would have the most impact and I used the basic scaler and a minmax scaler.

Then I ran some tests on all the models with default parameters and optimized parameters with and without preprocessing and compared the results.

I then output the results sorting by accuracy first and time second then labeling the best one.

Results:



```
List of algorithms sorted best to worst:

Random Forest HPO ('scaler'): 69.48980% accuracy, 20.74624 seconds
Random Forest HPO ('MMScaler'): 69.08163% accuracy, 18.44980 seconds
Random Forest Default ('MMScaler'): 68.57143% accuracy, 0.91583 seconds
Random Forest HPO (No Preprocessing): 68.57143% accuracy, 19.88811 seconds
Random Forest Default ('scaler'): 68.46939% accuracy, 0.85678 seconds
Random Forest Default (No Preprocessing): 67.95918% accuracy, 0.85878 seconds
Support Vector Machine HPO ('scaler'): 60.51020% accuracy, 20.51067 seconds
Support Vector Machine HPO ('MMScaler'): 58.06122% accuracy, 19.79302 seconds
K-Nearest-Neighbors Default ('scaler'): 57.75510% accuracy, 0.00901 seconds
K-Nearest-Neighbors HPO ('scaler'): 56.73469% accuracy, 6.76816 seconds
K-Nearest-Neighbors Default ('MMScaler'): 55.91837% accuracy, 0.00901 seconds
Support Vector Machine Default ('scaler'): 55.91837% accuracy, 0.40137 seconds
K-Nearest-Neighbors HPO ('MMScaler'): 55.91837% accuracy, 7.06643 seconds
Support Vector Machine Default ('MMScaler'): 55.20408% accuracy, 0.39436 seconds
K-Nearest-Neighbors HPO (No Preprocessing): 48.87755% accuracy, 5.82660 seconds
Search Vector Machine HPO (No Preprocessing): 48.77551% accuracy, 23.78937 seconds
K-Nearest-Neighbors Default (No Preprocessing): 48.06122% accuracy, 0.00701 seconds
Search Vector Machine Default (No Preprocessing): 44.38776% accuracy, 0.42439 seconds
Best Algorithm by accuracy and time is: Random Forest HPO ('scaler')
```

The results show some improvements for random forest with preprocessed data and huge improvements for the support vector and k nearest neighbors algorithms when the data is preprocessed.

Code:

Main.py