Introduction: There are many machine learning algorithms to choose from when fitting a dataset. In this exercise, three different ML algorithms were explored to find the best algorithm that fits the chosen model in absence of over-fitting.

Here, the penguin size data set was used to predict the body mass of different penguins investigating three different ML algorithms: Random Forest, SVM, and Linear Regression. Hyperparameters were modified in hopes of avoiding over-fitting.

Dataset Description: In this data set, penguin size is measured by seven features including species, island, culmen length, culmen depth, flipper length, body mass, and sex. For each of these features, the number of observations was unequal. Thus, to equalize the number of observations between all features, I used a drop null values function "drop null values" to assess this. After this function was performed , the number of observations for each feature was 333. Next, I used the information command to asses the data type. Here, data type was not consistent throughout each feature with species, island, and sex features showing the object data type. Here, I used the replace function to replace each object term with a number instead. For species, Adelie=1, Chinstrap=2, and Gentoo=3. For island, Torgersen=1, Biscoe=2, Dream=3. For sex, male=1 and female=2. Next, I used a get_dummies function to convert categorical variables to indictor variables.

Experimental Setup: To explore different ML algorithms, I used Python as the programming language along with a variety of packages. These packages include pandas and numpy for data manipulation and array building. Sklearn was also used for statistical modeling such as Random Forest, Randomized Search CV, Grid Search CV, SVM, SVC, and Linear Regression. For the Penguin Size data set, I compared three different algorithms that included Random Forest, SVM, and Linear Regression. To evaluate each, I optimized hyperparameters and evaluated an algorithm score. Additional statistical tests were run to assess overfitting.

Results: Upon testing three algorithms, a performance measure and overfitting statistic were generated for each. Additionally, data was split into testsing and training data sets using train_test_split from sklearn. The test size for all models was 20% and train size was 80%.

Beginning with Random Forest, before changing hyperparameters, the baseline error for the penguin data set was 3984.88. This baseline data was used as a starting place measure that can be compared to the final model measure. Here, the final model should perform better than the baseline. To set the baseline, I chose to use the feature flipper length as the baseline prediction. Then, I measured the performance of my baseline prediction by calculating the baseline error and average baseline error.

The mean baseline error was 227.6 mm, which stemmed from the metric used for the baseline prediction (flipper length mm). The mean baseline error here is a measure of the difference between the baseline prediction and the test y data. The average baseline error is the mean absolute error of the baseline prediction.

Next, since I aim to predict body mass, I determined how well my model was performing by calculating an accuracy. To do this, I first calculated the mean absolute percent error (MAPE). MAPE takes the sum of the absolute value of actual values minus predicted values divided by actual values with all of this divided by the total number of points or samples. Once this was calculated, accuracy was measured by subtracting the mean MAPE from 100. The prediction accuracy was 94.41%.

From here, a random search and grid search were performed. The hyperparamters that were tuned included max_depth, max_depth.append, min_samples_split and min_samples_leaf. This concluded in finding hyperparameters that resulted in a 0.67% improvement.

Finally, an SVC classifier score for train and test data was assessed to look at overfitting. This score calculates the mean accuracy of the inputted test and label data. Here, the train score was 0.04 and the tests score was 0.02. The train score is larger than the tests score suggesting over-fitting is not of high risk.

Next, the SVM algorithm was performed. Once the model was fit, the model

score was 4.35%. This score calculates the mean accuracy of the inputted test and label data. Next, a grid search was performed and a classification report was generated after the hyperparameters were chosen. Here, the parameters that were tuned include C, gamma, and kernel. After choosing hyperparameters, accuracy was 6%. Lastly, to check for overfitting scores of the train and test data was observed. Here, the training score was 0.11 and the testing score was 0.04. The train score is larger than the tests score suggesting over-fitting is not of high risk.

Lastly, a linear regression algorithm was performed. Before changing hyperparameters, the algorithm score was 0.87. This score shows the the coefficient of determination of the prediction. After manually changing different hyperparameters, the algorithm score was 0.87. The parameters manually tuned included copy_X, fit_intercept, n_jobs, and positive. Then, the train score was 0.868 and the test score was 0.886. The train and test scores are very similar here, so over-fitting is of minimal risk.

References:

1. GfG. "Python Pandas - Get_dummies() Method." GeeksforGeeks, GeeksforGeeks, 13 Oct. 2020, www.geeksforgeeks.org/python-pandas-get_dummies-method/.
2. Koehrsen, Will. "Random Forest in Python." Medium, Towards Data Science, 17 Jan. 2018, towardsdatascience.com/random-forest-in-python-24d0893d51c0.
3. Simbarashe ZuvaSimbarashe Zuva 2377 bronze badges, and Arya McCarthyArya McCarthy 8. "Attributeerror: 'numpy.Ndarray' Object Has No Attribute 'Score' Error." Stack Overflow, 1 Jan. 1967, stackoverflow.com/questions/66318181/attributeerror-numpy-ndarray-object-has-no-attribute-score-error.
4. "1.1. Linear Models." Scikit, scikit-learn.org/stable/modules/linear_model.html. Accessed 7 Mar. 2024.
5. "Implementing Support Vector Machine (SVM) in Python." DatabaseTown, 14 Mar. 2023, databasetown.com/implementing-support-vector-machine-svm-in-python/.
6. "Pandas." Pandas, pandas.pydata.org/. Accessed 7 Mar. 2024.
7. "Sklearn.Svm.SVC." Scikit, scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed 14 Mar. 2024.