Introduction: In this report, we explore the performance of AutoSklearn, a tool that automates the process of selecting the best learning algorithm and hyperparameters for a given dataset. AutoSklearn utilizes meta-learning to learn from past training information and pick the optimal model configuration. However, in our initial experimentation, we found that the default parameters for Random Forest Classifier performed better than AutoSklearn. To address this issue, we delved deeper into our dataset and implemented various data preprocessing techniques to enhance AutoSklearn's performance.

Data Description: On exploring the data, we found that the dataset consists of the data of the wine class. The dataset provides valuable insights into the chemical properties and wine's quality ratings. It comprises of 12 different parameters named as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol and class.

Experimental Changes done: We began by examining the dataset using the '.info()' and '.describe()' functions. Additionally, we experimented with '.value_count()' function to observe the count and proportion of target feature in our dataset. Through this exploration, we discovered that the dataset was imbalanced, with certain classes being underrepresented. Imbalanced datasets pose a challenge for classification algorithms, as they may bias predictions towards the majority class. Therefore, we decided to address this issue before running AutoSklearn. To mitigate the effects of class imbalance, we employed the Synthetic Minority Oversampling Technique (SMOTE). SMOTE works by generating synthetic samples from the minority class to balance the dataset. We used the 'SMOTE' function from the 'imblearn' library to oversample the minority class and increase its representation in the dataset.

After applying SMOTE, we performed a train-test split with stratification. Stratification ensures an even distribution of target variable classes in both the training and testing sets, thereby preventing data leakage and bias in model evaluation. We then further explored the dataset to find out the correlated parameters in the datset and tried to remove them but there was no significant increase in the accuracy of Autosklearn.

After the above two steps, we tried changing the random state value in the experiment and noticed that the results were varying greatly with different random state values. We used 'np.random.random_state' function to randomly select the random state values and after running it for multiple times.

Results: Following the preprocessing steps, we retrained both the Random Forest Classifier and AutoSklearn Classifier. The default Random Forest Classifier achieved an accuracy of 0.84. However, after utilizing AutoSklearn, the accuracy improved to 0.86. This enhancement underscores the importance of proper data preprocessing in improving AutoSklearn's performance.

Initial Result	
Model	**Accuracy**
Random Forest	0.67

AutoSklearn	0.65

Final Result	
Model	**Accuracy**
Random Forest	0.84
AutoSklearn	0.86

References:

- Auto-Sklearn for Automated Machine Learning in Python https://machinelearningmastery.com/auto-sklearn-for-automated-machine-learning-in-python/
- Common pitfalls and anti-patterns that occur when using scikit-learn https://scikit-learn.org/stable/common_pitfalls.html
- Data fetching using openml https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html
- Handling imbalanced data https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/
- Stratified sampling in machine learning https://www.baeldung.com/cs/ml-stratified-sampling