Introduction: In this experiment, we aim to optimize the entire machine learning (ML) pipeline, including preprocessing steps and algorithm selection, using hyperparameter optimization techniques. The primary goal is to demonstrate the effectiveness of systematically tuning hyperparameters across multiple components of an ML pipeline to improve predictive performance. The problem we are addressing involves building robust ML models for classification tasks on two different datasets: the Wine Quality dataset and Eagleford drilling dataset. These datasets exhibit different characteristics in terms of features, observations, and target variables, providing diverse challenges for the optimization process. To solve this problem, we will employ a comprehensive experimental setup that includes various preprocessing techniques, machine learning algorithms, and hyperparameter optimization methods. We will be using Synthetic Minority Oversampling Technique (SMOTE) to mitigate the data imbalance problem as we already know from previous exercises that both our data sets have imbalanced proportion of target variable. The pipeline consists of feature selection, data normalization and a machine learning model for prediction.

Data description: The first data set consists of the drilling data gathered from one of the oil and gas operators in the Eagleford shale region in Texas (USA). The collected data consists of 17 different parameters named as Depth, Hook Load, Bit Weight, Block Height, Rate of Penetration (ROP), Top Drive RPM, Top Drive Torque, Differential Pressure, Flow In Rate, Pump SPM (Strokes per minute), Flow Out Percent, Bit Size, Gamma Ray, Mud Weight In, Pump Pressure, D-exponent and formation. The aim of this dataset is to predict the formation by using other available parameters as an input. From the previous exercises we already know that this dataset doesn’t have any null values.

The second dataset consists of the data of the white variant of the Portuguese "Vinho Verde" wine. 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 quality. The aim of this dataset is to predict the wine quality from other available input parameters. From the previous exercises we already know that this dataset as well doesn’t have any null values.

Experimental setup: The programming language used for this experiment will be python. We begin with importing a few libraries named as pandas, numpy. After this we imported SMOTE from imblearn.over_sampling. After this we uploaded the first dataset i.e. Eagleford drilling data using pd.read_csv function. We then divided the dataset into input and target variables where ‘formation’ was the target variable while all the other parameters were taken as input. After this we used value_counts() function to find out the value counts of the different ‘formations’ present in the dataset. As the dataset was found out to be imbalanced, we have now used Synthetic Minority Oversampling Technique, or SMOTE to balance the data. After this we begin with building our pipeline. We first begin with importing necessary libraries. We imported standard scalar from sklearn.preprocessing, RFE from sklearn.feature_selection, Random Forest classifier from sklearn.ensemble, pipeline from sklearn.pipeline, cross_val_score, kfold from sklearn.model_selection and accuracy_score from sklearn.metrices. In the initial scenario we build our pipeline without hyperparameter tuning. Here we have used k-5 cross validation. We then defined our pipeline. The pipeline consists of three components, standard scalar for normalizing the data, Recursive feature elimination (RFE) for feature selection and Random Forest classifier for classification. After this we ran the cross-validation loop and finally fitted the pipeline on X_train and y_train. Finally, we have calculated the mean accuracy score for this scenario and the same has been reported in the results section.

In the second scenario along with the above steps we have also performed hyperparameter tuning using random search for the pipeline and have also incorporated nested resampling with inner and outer splits being equal to 5. For this scenario we have imported RandomsearchCV from sklearn.model_selection, randint from scipy.stats along with other modules mentioned above. The search space for different hyperparameters is given as: 'n_features_to_select': randint(5, 20), 'n_estimators': randint(50, 200), 'max_depth': [None, 10, 20, 30], 'min_samples_split': randint(2, 20). After defining the hyperparameter search space we performed an outer cross validation loop for nested resampling with k=5 and then inner cross validation loop for hyperparameter tuning. The number of iterations tried for random search was 50 and 100. Then we have finally reported the mean accuracy score for this scenario as well.

After this we uploaded the second dataset which is the white wine quality dataset using pd.read_csv function. We then divided the dataset into input and target variables where ‘quality’ was the target variable while all the other parameters were taken as input. After this we used value_counts() function to find out the value counts of the different wine qualities present in the dataset. As the dataset was found out to be imbalanced, we have now used Synthetic Minority Oversampling Technique, or SMOTE to balance the data. After this we begin with building our pipeline. All the necessary libraries used have already been mentioned above. In the initial scenario we build our pipeline without hyperparameter tuning. Here we have used k-5 cross validation. We then defined our pipeline. The pipeline consists of three components, standard scalar for normalizing the data, Recursive feature elimination (RFE) for feature selection and Random Forest classifier for classification. After this we ran the cross-validation loop and finally fitted the pipeline on X1_train and y1_train. Finally, we have calculated the mean accuracy score for this scenario and the same has been reported in the results section.

In the second scenario for the wine quality dataset, along with the above steps we have also performed hyperparameter tuning using random search for the pipeline and have also incorporated nested resampling with inner and outer splits being equal to 5. For this scenario libraries and modules used remain the same as above. The search space for different hyperparameters is given as: 'n_features_to_select': randint(5, 20), 'n_estimators': randint(50, 200), 'max_depth': [None, 10, 20, 30], 'min_samples_split': randint(2, 20). After defining the hyperparameter search space we performed an outer cross validation loop for nested resampling with k=5 and then inner cross validation loop for hyperparameter tuning. The number of iterations tried for random search was 50 and 100. Then we have finally reported the mean accuracy score for this scenario as well.

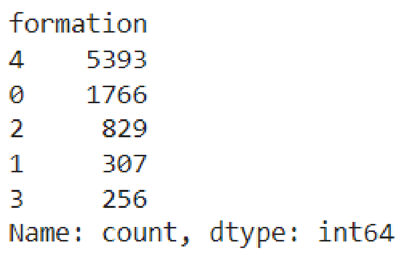
Results: The value_counts() function showed that both the datasets were imbalanced, and the values have been displayed in the appendix section. The mean accuracy obtained from the Eagle ford dataset without hyperparameter tuning was 0.98. The mean accuracy obtained with the same dataset with hyperparameter tuning with 50 and then 100 iterations was equal to 0.99. The mean accuracy obtained with the wine quality dataset without hyperparameter tuning was 0.88. The mean accuracy for wine quality dataset with both 50 and 100 iterations for random search hyperparameter tuning was 0.89.

References:

1) Sklearn pipeline https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
2) Feature selection RFE https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html
3) Modeling Pipeline Optimization With scikit-learn https://machinelearningmastery.com/modeling-pipeline-optimization-with-scikit-learn/
4) Sklearn ensemble Random Forest https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

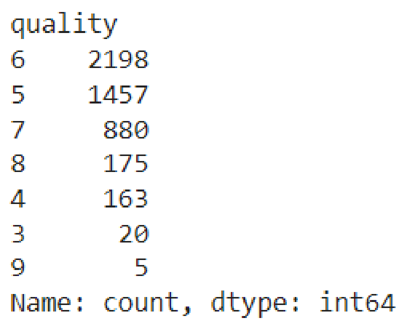
Appendix

Value counts for the differ5ent formations in the Eagleford dataset



```
formation
4    5393
0    1766
2     829
1     307
3     256
Name: count, dtype: int64
```

Value counts for the different wine quality in wine quality dataset.



```
quality
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: count, dtype: int64
```