Exercise 4.4: Random Forest with Class Imbalance

□□□ Random Forest with Class Imbalance

The goal of this exercise is to investigate the performance of *Random Forest* on a dataset with class imbalance, and then use corrections strategies to improve performance.

Your final comparison may look like the table below (but not with these exact values):

Instructions:

- Read the dataset `diabetes.csv` as a pandas dataframe.
- Take a quick look at the dataset.
- Split the data into train and test sets.
- Perform classification with a Vanilla Random Forest which does not take into account class imbalance.
- Perform classification with a Balanced Random Forest which does take into account class imbalance.
- Upsample the data and perform classification with a Balanced Random Forest.
- Downsample the data and perform classification with a Balanced Random Forest.
- Compare the F1-Score and AUC Score of all 4 models.

Hints:

```
np.ravel()
```

Return a contiguous flattened array.

`f1_score()`

Compute the F1 score, also known as balanced F-score or F-measure.

`roc_auc_score()`

Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.

`sklearn.train_test_split()`

Split arrays or matrices into random train and test subsets.

`RandomForestClassifier()`

Defines the RandomForestClassifier and includes more details on the definition and range of values for its tunable parameters.

`RandomForestClassifier.fit()`

Build a forest of trees from the training set (X, y).

`RandomForestClassifier.predict()`

Predict class for X.

`BalancedRandomForestClassifier()`

A balanced random forest classifier.

`BalancedRandomForestClassifier.fit()`

Build a forest of trees from the training set (X, y).

`BalancedRandomForestClassifier.predict()`

Predict class for X.

`SMOTE()`

Class to perform over-sampling using SMOTE.

`SMOTE.fit_resample()`

Resample the dataset.

`RandomUnderSampler()`

Class to perform random under-sampling.

```
RandomUnderSampler.fit_resample()
```

Resample the dataset.