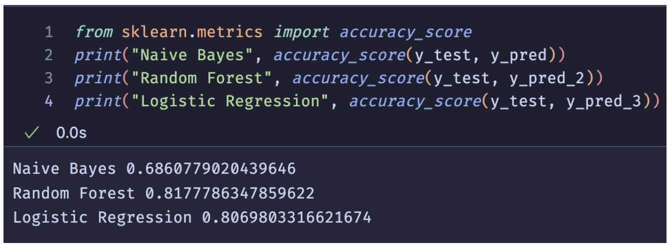
Hate Speech Detection - scikit-learn

Ravi Regulagedda, Zackary Leech

Text pre-processing was performed on the dataset, following which it was vectorized by using CountVectorizer from sci-kit learn. Following this the vectors were scaled using StandardScaler and then three different basic classification models were tested - Logistic Regression, Naive Bayes and Random Forest. Their base model accuracy scores are given below -

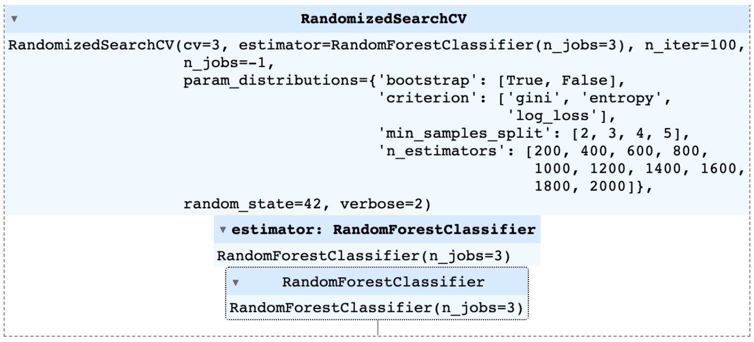


```
from sklearn.metrics import accuracy_score
print("Naive Bayes", accuracy_score(y_test, y_pred))
print("Random Forest", accuracy_score(y_test, y_pred_2))
print("Logistic Regression", accuracy_score(y_test, y_pred_3))
✓ 0.0s

Naive Bayes 0.6860779020439646
Random Forest 0.8177786347859622
Logistic Regression 0.8069803316621674
```

Figure 1: Result showing the baseline accuracy of 3 different algorithms

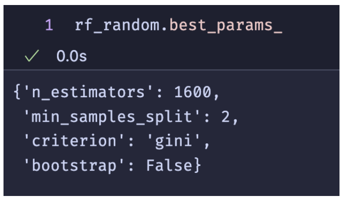
Following this, a cross-validation search was performed on the ranges of hyper parameters for Random Forest since it had the highest initial accuracy. The ranges chosen are below



```
RandomizedSearchCV
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(n_jobs=3), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'criterion': ['gini', 'entropy',
                                                      'log_loss'],
                                        'min_samples_split': [2, 3, 4, 5],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   random_state=42, verbose=2)
estimator: RandomForestClassifier
RandomForestClassifier(n_jobs=3)
RandomForestClassifier
RandomForestClassifier(n_jobs=3)
```

Figure 2: Range of hyperparameter values for CV search

We used a 3 fold cross validation over 100 searches making a total of 300 models fit. This provided the following values for the hyperparameters -

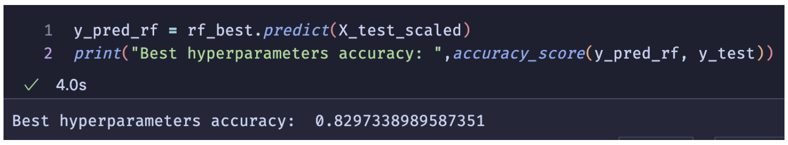


```
rf_random.best_params_
✓ 0.0s

{'n_estimators': 1600,
 'min_samples_split': 2,
 'criterion': 'gini',
 'bootstrap': False}
```

Figure 3: Range of hyperparameter values for CV search

However, many of the “optimal” values provided are just the default values of the parameters. Specifically, *min_samples_split* which is the minimum number of positive/negative examples required to split a leaf, *criterion* which is the way information at each node is calculated and *bootstrap*, which is whether to use a bootstrapped dataset of the entire set are all at their default values. Therefore, the increase in accuracy was just 1%. The final accuracy is shown below -



```
y_pred_rf = rf_best.predict(X_test_scaled)
print("Best hyperparameters accuracy: ",accuracy_score(y_pred_rf, y_test))
```

```
Best hyperparameters accuracy:  0.8297338989587351
```

Figure 4: Accuracy with best hyperparameters