Ungraded lab: Permutation Feature Importance

Welcome, during this ungraded lab you are going to be perform Permutation Feature Importance on the [wine dataset]() using scikit-learn. In particular you will:

1. Train a Random Forest classifier on the data.
2. Compute the feature importance score by permutating each feature.
3. Re-train the model with only the top features.
4. Check other classifiers for comparison.

Let's get started!

Inspect and pre-process the data

Begin by upgrading scikit-learn to the latest version:

```
!pip install -U scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (0.22.2.post1)
Collecting scikit-learn
  Downloading scikit_learn-0.24.2-cp37-cp37m-manylinux2010_x86_64.whl (22.3 MB)
     |████████████████████████████████| 22.3 MB 1.2 MB/s
Collecting threadpoolctl>=2.0.0
  Downloading threadpoolctl-2.2.0-py3-none-any.whl (12 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-lear
Requirement already satisfied: scipy>=0.19.1 in /usr/local/lib/python3.7/dist-packages (from scikit-lea
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scikit-lea
Installing collected packages: threadpoolctl, scikit-learn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.22.2.post1
    Uninstalling scikit-learn-0.22.2.post1:
      Successfully uninstalled scikit-learn-0.22.2.post1
Successfully installed scikit-learn-0.24.2 threadpoolctl-2.2.0
```

Now import the required dependencies and load the dataset:

```
import numpy as np
from sklearn.datasets import load_wine

# as_frame param requires scikit-learn >= 0.23
data = load_wine(as_frame=True)

# Print first rows of the data
data.frame.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenol
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.2
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.2
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.3
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.2
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.3

This dataset is made up of 13 numerical features and there are 3 different classes of wine.

Now perform the train/test split and normalize the data using [`StandardScaler`]():

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Train / Test split
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, random_state=42)

# Instantiate StandardScaler
scaler = StandardScaler()
```

```
# Fit it to the train data
scaler.fit(X_train)

# Use it to transform the train and test data
X_train = scaler.transform(X_train)

# Notice that the scaler is trained on the train data to avoid data leakage from the test set
X_test = scaler.transform(X_test)
```

Train the classifier

Now you will fit a [Random Forest classifier] with 10 estimators and compute the mean accuracy achieved:

```
from sklearn.ensemble import RandomForestClassifier

# Fit the classifier
rf_clf = RandomForestClassifier(n_estimators=10, random_state=42).fit(X_train, y_train)

# Print the mean accuracy achieved by the classifier on the test set
rf_clf.score(X_test, y_test)
```

```
0.9111111111111111
```

This model achieved a mean accuracy of 91%. Pretty good for a model without fine tunning.

Permutation Feature Importance

To perform the model inspection technique known as Permutation Feature Importance you will use scikit-learn's built-in function `permutation_importance`.

You will create a function that given a classifier, features and labels computes the feature importance for every feature:

```
from sklearn.inspection import permutation_importance

def feature_importance(clf, X, y, top_limit=None):

  # Retrieve the Bunch object after 50 repeats
  # n_repeats is the number of times that each feature was permuted to compute the final score
  bunch = permutation_importance(clf, X, y,
                                 n_repeats=50, random_state=42)

  # Average feature importance
  imp_means = bunch.importances_mean

  # List that contains the index of each feature in descending order of importance
  ordered_imp_means_args = np.argsort(imp_means)[::-1]

  # If no limit print all features
  if top_limit is None:
    top_limit = len(ordered_imp_means_args)

  # Print relevant information
  for i, _ in zip(ordered_imp_means_args, range(top_limit)):
```

```
Feature color_intensity with index 9 has an average importance score of 0.112 +/- 0.023

Feature od280/od315_of_diluted_wines with index 11 has an average importance score of 0.007 +/- 0.005

Feature total_phenols with index 5 has an average importance score of 0.003 +/- 0.004

Feature malic_acid with index 1 has an average importance score of 0.002 +/- 0.004

Feature proanthocyanins with index 8 has an average importance score of 0.002 +/- 0.003

Feature hue with index 10 has an average importance score of 0.002 +/- 0.003

Feature nonflavanoid_phenols with index 7 has an average importance score of 0.000 +/- 0.000

Feature magnesium with index 4 has an average importance score of 0.000 +/- 0.000

Feature alcalinity_of_ash with index 3 has an average importance score of 0.000 +/- 0.000

Feature ash with index 2 has an average importance score of 0.000 +/- 0.000

Feature alcohol with index 0 has an average importance score of 0.000 +/- 0.000
```

Looks like many of the features have a fairly low importance score. This points that the predictive power of this dataset is conmdensed in a few features.

However it is important to notice that this process was done for the training set, so this feature importance does NOT have into account if the feature might help with the generalization power of the model.

To check this, repeat the process for the test set:

```
feature_importance(rf_clf, X_test, y_test)
```

```
Feature flavanoids with index 6 has an average importance score of 0.202 +/- 0.047

Feature proline with index 12 has an average importance score of 0.143 +/- 0.042

Feature color_intensity with index 9 has an average importance score of 0.112 +/- 0.043

Feature alcohol with index 0 has an average importance score of 0.024 +/- 0.017

Feature magnesium with index 4 has an average importance score of 0.021 +/- 0.015

Feature od280/od315_of_diluted_wines with index 11 has an average importance score of 0.015 +/- 0.018
```

```
print("\nOn TEST split:\n")
feature_importance(rf_clf, X_test, y_test, top_limit=3)
```

```
On TRAIN split:

Feature flavanoids with index 6 has an average importance score of 0.227 +/- 0.025

Feature proline with index 12 has an average importance score of 0.142 +/- 0.019

Feature color_intensity with index 9 has an average importance score of 0.112 +/- 0.023


On TEST split:

Feature flavanoids with index 6 has an average importance score of 0.202 +/- 0.047

Feature proline with index 12 has an average importance score of 0.143 +/- 0.042

Feature color_intensity with index 9 has an average importance score of 0.112 +/- 0.043
```

```
# Preserve only the top 3 features
X_train_top_features = X_train[:,[6, 9, 12]]
X_test_top_features = X_test[:,[6, 9, 12]]

# Re-train with only these features
rf_clf_top = RandomForestClassifier(n_estimators=10, random_state=42).fit(X_train_top_features, y_train
```

```
        "Support Vector": SVC()}


# Compute feature importance on the test set given a classifier
def fit_compute_importance(clf):
  clf.fit(X_train, y_train)
  print(f"📏 Mean accuracy score on the test set: {clf.score(X_test, y_test)*100:.2f}%\n")
  print("🔝 Top 4 features when using the test set:\n")
  feature_importance(clf, X_test, y_test, top_limit=4)
```