Tasks for labs:

- instead of face recognition, use parking lot dataset (1. exercise):
  - http://mrl.cs.vsb.cz//people/fusek/ano2_course.html
- visualize several best features:

- create parking lot occupancy recognizer with the use of Haar features combined with RandomForest or AdaBoost

- in addition to this, try to experiment with HOG and LBP Features and try to compare recognition performances (Haar vs. HOG vs. LBP) - with the use of scikit-image and opencv library

- for visualization of results (comparison) try to use python libraries (e.g. Matplotlib, Seaborn)

Instead of the adaboost classifier, the RandomForest classifier is used in this sklearn example.

```
from time import time

import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

from skimage.data import lfw_subset
from skimage.transform import integral_image
from skimage.feature import haar_like_feature
from skimage.feature import haar_like_feature_coord
from skimage.feature import draw_haar_like_feature
```

https://scikit-image.org/docs/stable/auto_examples/applications/plot_haar_extraction_selection_classification.html

The procedure to extract the Haar-like features from an image is relatively simple. Firstly, a region of interest (ROI) is defined. Secondly, the integral image within this ROI is computed. Finally, the integral image is used to extract the features.

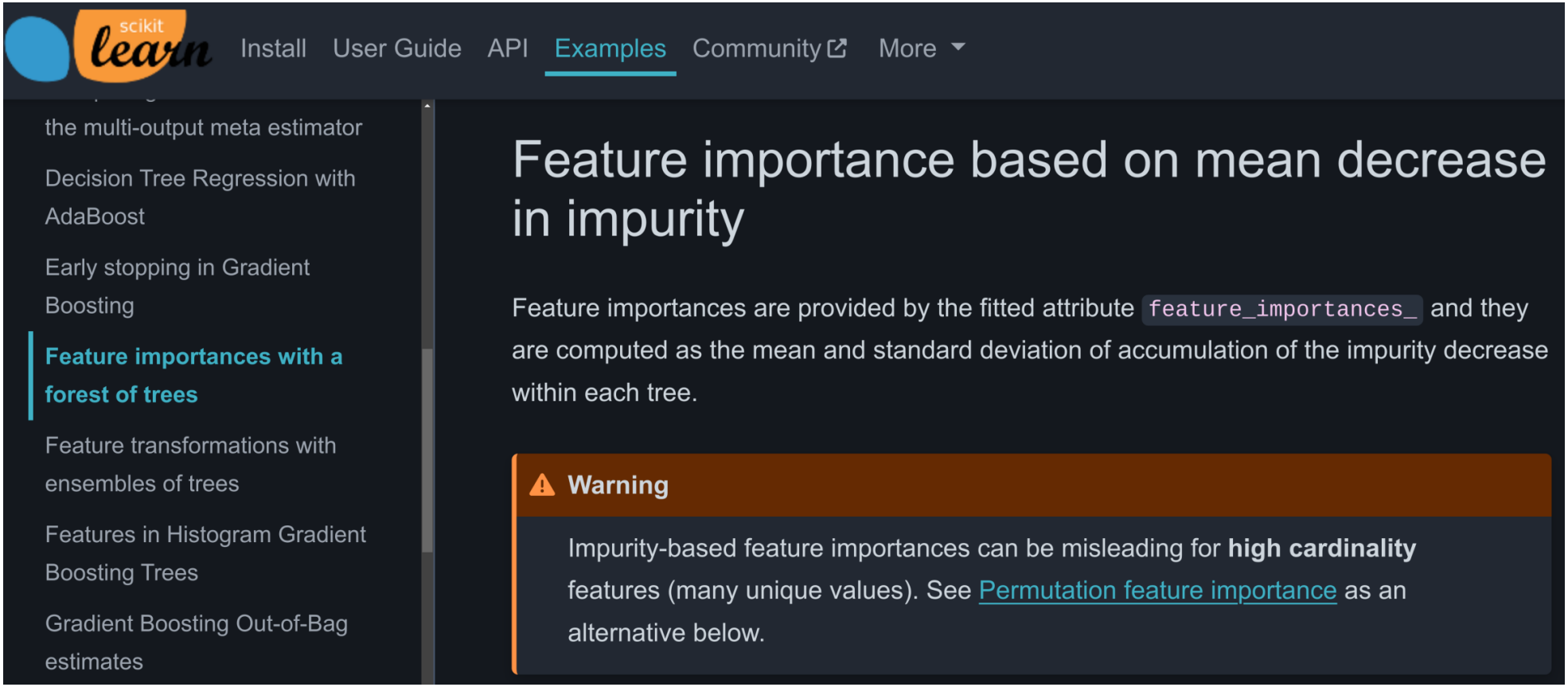
```
def extract_feature_image(img, feature_type, feature_coord=None):
    """Extract the haar feature for the current image"""
    ii = integral_image(img)
    return haar_like_feature(
        ii,
        0,
        0,
        ii.shape[0],
        ii.shape[1],
        feature_type=feature_type,
        feature_coord=feature_coord,
    )
```

https://scikit-image.org/docs/stable/auto_examples/applications/plot_haar_extraction_selection_classification.html

A random forest classifier can be trained in order to select the most salient features, specifically for face classification. The idea is to determine which features are most often used by the ensemble of trees. By using only the most salient features in subsequent steps, we can drastically speed up the computation while retaining accuracy.

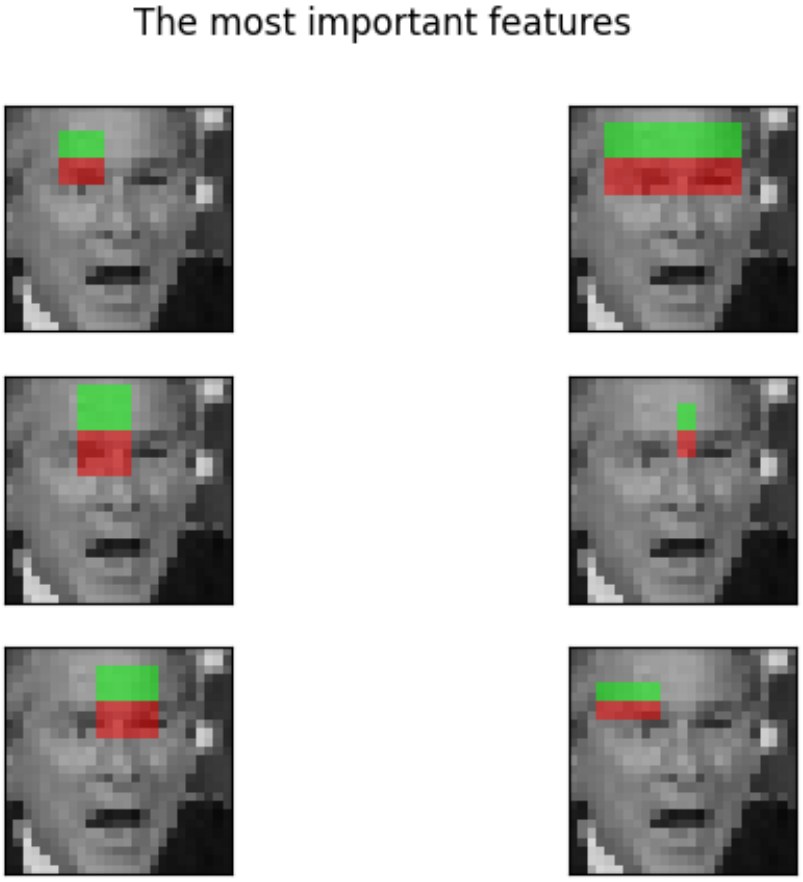
```
# Train a random forest classifier and assess its performance
clf = RandomForestClassifier(
    n_estimators=1000, max_depth=None, max_features=100, n_jobs=-1, random_state=0
)
t_start = time()
clf.fit(X_train, y_train)
time_full_train = time() - t_start
auc_full_features = roc_auc_score(y_test, clf.predict_proba(X_test)[:, 1])

# Sort features in order of importance and plot the six most significant
idx_sorted = np.argsort(clf.feature_importances_)[::-1]

fig, axes = plt.subplots(3, 2)
for idx, ax in enumerate(axes.ravel()):
    image = images[0]
    image = draw_haar_like_feature(
        image, 0, 0, images.shape[2], images.shape[1], [feature_coord[idx_sorted[idx
    )
    ax.imshow(image)
    ax.set_xticks([])
    ax.set_yticks([])

_ = fig.suptitle('The most important features')
```

https://scikit-image.org/docs/stable/auto_examples/applications/plot_haar_extraction_selection_classification.html



https://scikit-image.org/docs/stable/auto_examples/applications/plot_haar_extraction_selection_classification.html

https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html



https://scikit-image.org/docs/stable/auto_examples/applications/plot_haar_extraction_selection_classification.html