Lecture 5 - Student Notebook

We first load and clean the data.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, cross_validate, GridSearchCV, ParameterGrid
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report, confusion_matrix, auc
from sklearn.utils import resample

DATA_DIR = "./../../data"

# Parse the aggregated student data frame.
# This data is from an EPFL Linear Algebra flipped classroom. df_lq is aggregated features for the last week of student performance.
# ts represents the students' time series features.

df_lq = pd.read_csv('{}/aggregated_extended_fc.csv'.format(DATA_DIR))
ts = pd.read_csv('{}/time_series_extended_fc.csv'.format(DATA_DIR))

def remove_inactive_students(df, ts):
    """
    Filter the students (removing the ones that are inactive) to proceed with analysis on students who have participated during the entire class.
    Inputs: df, ts
    Outputs: filtered df, ts
    """
    # Fill all NaNs with strings to make them easier to process
    df = df.fillna('NaN')

    # Find all users weeks with 0 clicks on weekends and 0 clicks on weekdays during the first weeks of the semester
    df_first = ts[ts.week < 5]
    rows = np.where(np.logical_and(df_first.ch_total_clicks_weekend==0, df_first.ch_total_clicks_weekday==0).to_numpy())[0]
    df_zero = df_first.iloc[rows, :]
    dropusers = np.unique(df_zero.user)

    # Drop users with no activity
    ts = ts[~ts.user.isin(dropusers)]
```

```
    df = df[~df.user.isin(dropusers)]
    return df, ts

df_lq, ts = remove_inactive_students(df_lq, ts)
```

The `compute_scores` function computes the performance of classifiers with accuracy + AUC. We will use this evaluation function for all our experiments.

```
def compute_scores(clf, X_train, y_train, X_test, y_test, roundnum=3,
report=False):
    """
    Train clf (binary classification) model on X_train and y_train,
predict on X_test. Evaluate predictions against ground truth y_test.
    Inputs: clf, training set (X_train, y_train), test set (X_test,
y_test)
    Inputs (optional): roundnum (number of digits for rounding
metrics), report (print scores)
    Outputs: accuracy, AUC
    """
    # Fit the clf predictor (passed in as an argument)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)

    # Calculate roc AUC score
    AUC = roc_auc_score(y_test, clf.predict_proba(X_test)[:, 1])

    # Print classification results
    if report:
        print(classification_report(y_test, y_pred))

    return round(accuracy, roundnum), round(AUC, roundnum)
```

We compute the pass/fail label of the students in the dataframe to use for the experiments. We will use the aggregated dataframe (df_lq) for all our experiments. If students have a grade higher than or equal to 4, they have passed the class.

```
df_lq['passed'] = (df_lq.grade >= 4).astype(int)
```

We are interested in model selection and assessment. We will use a random forest model for all our evaluations. For our evaluations, we will investigate behavioral features only.

```
# Filter out demographic features
features = [x for x in df_lq.columns if x not in ['user', 'week',
'grade', 'gender', 'category', 'year', 'passed']]
print(features)
```

```
['ch_num_sessions', 'ch_time_in_prob_sum', 'ch_time_in_video_sum',
'ch_ratio_clicks_weekend_day', 'ch_total_clicks_weekend',
```

```
'ch_total_clicks_weekday', 'ch_time_sessions_mean',
'ch_time_sessions_std', 'bo_delay_lecture', 'bo_reg_peak_dayhour',
'bo_reg_periodicity_m1', 'ma_competency_strength',
'ma_competency_anti', 'ma_content_anti', 'ma_student_shape',
'ma_student_speed', 'mu_speed_playback_mean',
'mu_frequency_action_relative_video_pause', 'wa_num_subs',
'wa_num_subs_correct', 'wa_num_subs_avg', 'wa_num_subs_perc_correct',
'la_pause_dur_mean', 'la_seek_len_std', 'la_pause_dur_std',
'la_time_speeding_up_mean', 'la_time_speeding_up_std',
'la_weekly_prop_watched_mean', 'la_weekly_prop_interrupted_mean',
'la_weekly_prop_interrupted_std', 'la_weekly_prop_replayed_mean',
'la_weekly_prop_replayed_std', 'la_frequency_action_video_play']

# Only keep behavioral features in X.
X = df_lq[features]

# Our binary indicator variable is based on our evaluation criteria:
pass/fail.
y = df_lq['passed']
```

Your Turn 1: Model Assessment

In a first experiment, we are interested in assessing the generalizability of the trained model on to new data. We use two different methods to do so: a train-test split and a cross validation. Run the two methods and assess their accuracy/AUC:

- What can you observe?
- Where do the differences come from?

Train-Test Split

We split the data in a train-test split (stratified by the outcome variable) and obtain the accuracy and AUC.

```
# The train-test split is 80:20 (as shown by the 0.2 test_size
argument).
# We choose a random_state to replicate the results in the same split
every time we run this notebook.
# The stratify argument ensures a proportionate number of passes/fails
are in the training set and the test set.

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42, stratify=y)

# Let's initialize a RandomForestClassifier to make our model
predictions.

clf = RandomForestClassifier(random_state=42)

# We can use our compute_scores function to evaluate the results of
our train-test split classifier.
```

```
accuracy, AUC = compute_scores(clf, X_train, y_train, X_test, y_test)

print(f'Accuracy for train-test setting: {accuracy}')
print(f'AUC for train-test setting: {AUC}')
```

```
Accuracy for train-test setting: 0.723
AUC for train-test setting: 0.723
```

Cross Validation

We use a 10-fold cross validation to obtain accuracy and AUC.

```
# Initialize a new Random Forest predictor for our cross-validation
comparison.

clf = RandomForestClassifier(random_state=42)

# With the cross_validate function, the SciKit Learn library
automatically uses stratification across folds with the "cv" argument.

# In the background, it's using the StratifiedKFold function with 10
folds.
# We pass in our desired metrics ("accuracy", "roc_auc") for
evaluation in the "scoring" argument.

scores = cross_validate(clf, X, y, cv=3, scoring=['accuracy',
'roc_auc'])

print(f'Mean accuracy with cross-validation:
{scores["test_accuracy"].mean():.3f}')
print(f'Mean AUC with cross-validation:
{scores["test_roc_auc"].mean():.3f}')
```

```
Mean accuracy with cross-validation: 0.667
Mean AUC with cross-validation: 0.660
```

Your solution 1

Write your answers in the following cell:

```
import requests

exec(requests.get("https://courdier.pythonanywhere.com/get-send-
code").content)

npt_config = {
    'session_name': 'lecture-05',
    'session_owner': 'mlbd',
    'sender_name': input("Your name: "),
}
```

```
# YOUR TURN: what differences can you observe in the metrics
(accuracy, AUC) between train-test and cross validation? Where do
these differences come from?

### Share the answer with us
cv_tts = ""
send(cv_tts, 1)
```

```
Your name:  Paola

<Response [200]>
```

Your Turn 2: Model Selection

Of course, when training ML models, we want to tune their hyperparameters in order to optimize the performance. In order to tune the hyperparameters of a model, we need to do further splits of our data set. In the following, we present an incorrect example. Your task is to:

- Explain why it is incorrect.
- Describe how it could be fixed.

```
# We compute a grid search across the following parameter space
parameters = {
    'n_estimators': [20, 50, 100],
    'criterion': ['entropy', 'gini'],
    'max_depth': np.arange(3, 7),
    'min_samples_split': [2],
    'min_samples_leaf': [1],
}

# Perform 10-fold cross-validation to identify the best
hyperparameters, selecting the ones with the highest accuracy
clf = GridSearchCV(RandomForestClassifier(random_state=42),
parameters, cv=10, scoring=['accuracy', 'roc_auc'], refit='accuracy')
clf.fit(X, y)
```

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=42),
             param_grid={'criterion': ['entropy', 'gini'],
                         'max_depth': array([3, 4, 5, 6]),
                         'min_samples_leaf': [1], 'min_samples_split':
[2],
                         'n_estimators': [20, 50, 100]},
             refit='accuracy', scoring=['accuracy', 'roc_auc'])
```

```
clf.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 3,
 'min_samples_leaf': 1,
```

```
 'min_samples_split': 2,
 'n_estimators': 50}
```

```python
accuracy = clf.cv_results_['mean_test_accuracy'][clf.best_index_]
AUC = clf.cv_results_['mean_test_roc_auc'][clf.best_index_]

print(f'Accuracy for train-validation-test setting: {accuracy:.3f}')
print(f'AUC for train-validation-test setting: {AUC:.3f}')
```

```
Accuracy for train-validation-test setting: 0.726
AUC for train-validation-test setting: 0.707
```

Your Solution 2

```python
# YOUR TURN: Explain why it is incorrect.
answer = ""
send(answer, 2)
```

```
<Response [200]>
```

```python
# YOUR TURN: Describe how it could be fixed.
answer = ""
send(answer, 3)
```

```
<Response [200]>
```