# Assignment 1: SAL Machine Learning

## CYBR 5320

### Author: Hussain Quadri

This report will focus on comparing the metrics between CTU Netflow classification and validation methods: Namely KFolds Cross Validation (Using 2 folds with Random Forest Classifier in the interest of time constraints), and Leave One Out Crossvalidation (LOOCV).

The metrics we will focus on are:

- Precision
- Recall
- F1
- Accuracy
- ROC_AUC

In [20]:
```
from sklearn.model_selection import cross_validate
from sklearn.model_selection import KFold, StratifiedKFold
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Function to load the feature data
def load_features(file_path):
    with open(file_path, 'r') as f:
        features = np.loadtxt(f, delimiter=',')
    return features
```

In [26]:
```
def plot_result(x_label, y_label, plot_title, k_fold_data, loocv_data):

        # Set size of plot
        plt.figure(figsize=(12,6))
        labels = ["Precision", "Recall", "F1", "Accuracy", "ROC_AUC"]
        X_axis = np.arange(len(labels))
        ax = plt.gca()
        plt.ylim(0.20000, 1)
        plt.bar(X_axis-0.2, k_fold_data, 0.4, color='blue', label='KFold Score')
        plt.bar(X_axis+0.2, loocv_data, 0.4, color='red', label='LOOCV Score')
        plt.title(plot_title, fontsize=30)
        plt.xticks(X_axis, labels)
        plt.xlabel(x_label, fontsize=14)
        plt.ylabel(y_label, fontsize=14)
        plt.legend()
        plt.grid(True)
        plt.show()
```

In [11]:

```python
# Function to train/evaluate model using Stratified KFold cross validation
def stratified_kfold(files):

    # Load the feature data from each file and concatenate them - incorrect method.
    # features = np.concatenate([load_features(file) for file in files])
    # labels = np.concatenate([np.zeros(features.shape[0] // 2), np.ones(features.shape[0] // 2)])

    mean_precisions = []
    mean_recalls = []
    mean_f1s = []
    mean_accuracies = []
    mean_aucs = []
    num_iter = 1

    for file in files:
        features = load_features(file)

        # labels = np.concatenate([np.zeros(features.shape[0] // 2), np.ones(features.shape[0] // 2)])
        # Splitting using the features and labels above was producing better results
        # However, I ran into issues with Scenario 6 where the split size did not match and blocked progress
        # Hence I switched to the following method to split data



        X = features[:, 1:]
        y = features[:, 0]

        my_classifier = RandomForestClassifier(n_estimators=10, random_state=123)
        kf = StratifiedKFold(n_splits=2, shuffle=True, random_state=123)

        # Define arrays to store the evaluation metrics for each fold
        precisions = np.zeros(2)
        recalls = np.zeros(2)
        f1s = np.zeros(2)
        accuracies = np.zeros(2)
        aucs = np.zeros(2)

        # Iterate over the folds and train/evaluate the classifier
        for i, (train_index, test_index) in enumerate(kf.split(X, y)):
            X_train, X_test = X[train_index], X[test_index]
            y_train, y_test = y[train_index], y[test_index]

            my_classifier.fit(X_train, y_train)
            y_pred = my_classifier.predict(X_test)
            y_prob = my_classifier.predict_proba(X_test)[:, 1]

            precisions[i] = precision_score(y_test, y_pred)
            recalls[i] = recall_score(y_test, y_pred)
            f1s[i] = f1_score(y_test, y_pred)
            accuracies[i] = accuracy_score(y_test, y_pred)
            aucs[i] = roc_auc_score(y_test, y_prob)

        print(f'CTU Scenario: {num_iter}')
        print(f'\tAverage precision: {np.mean(precisions):.4f}')
        print(f'\tAverage recall: {np.mean(recalls):.4f}')
        print(f'\tAverage F1: {np.mean(f1s):.4f}')
        print(f'\tAverage accuracy: {np.mean(accuracies):.4f}')
        print(f'\tAverage AUC of ROC: {np.mean(aucs):.4f}\n\n')

        mean_precisions.append(np.mean(precisions))
        mean_recalls.append(np.mean(recalls))
        mean_f1s.append(np.mean(f1s))
        mean_accuracies.append(np.mean(accuracies))
        mean_aucs.append(np.mean(aucs))

        num_iter += 1

    # Return the average of the evaluation metrics across all folds
    return mean_precisions, mean_recalls, mean_f1s, mean_accuracies, mean_aucs
```

In [12]:

```
filename = 'features.txt'
files = []
for i in range(1, 14):
    files.append(str(i) + '/' + filename)

precision, recall, f1, accuracy, auc = stratified_kfold(files)
```

```
CTU Scenario: 1
        Average precision: 1.0000
        Average recall: 0.9990
        Average F1: 0.9995
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9999


CTU Scenario: 2
        Average precision: 1.0000
        Average recall: 0.9985
        Average F1: 0.9992
        Average accuracy: 1.0000
        Average AUC of ROC: 1.0000


CTU Scenario: 3
        Average precision: 1.0000
        Average recall: 0.9982
        Average F1: 0.9991
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9998


CTU Scenario: 4
        Average precision: 0.9996
        Average recall: 0.9922
        Average F1: 0.9959
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9977


CTU Scenario: 5
        Average precision: 0.9988
        Average recall: 0.9312
        Average F1: 0.9638
        Average accuracy: 0.9995
        Average AUC of ROC: 0.9961


CTU Scenario: 6
        Average precision: 1.0000
        Average recall: 0.9968
        Average F1: 0.9984
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9998


CTU Scenario: 7
        Average precision: 1.0000
        Average recall: 0.3014
        Average F1: 0.4631
        Average accuracy: 0.9996
        Average AUC of ROC: 0.8960


CTU Scenario: 8
        Average precision: 0.9998
        Average recall: 0.9803
        Average F1: 0.9899
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9987


CTU Scenario: 9
        Average precision: 0.9999
        Average recall: 0.9995
        Average F1: 0.9997
        Average accuracy: 0.9999
        Average AUC of ROC: 1.0000


CTU Scenario: 10
        Average precision: 1.0000
        Average recall: 0.9995
        Average F1: 0.9997
        Average accuracy: 1.0000
        Average AUC of ROC: 1.0000


CTU Scenario: 11
        Average precision: 1.0000
        Average recall: 0.9979
        Average F1: 0.9990
        Average accuracy: 0.9998
        Average AUC of ROC: 0.9996
```

```
CTU Scenario: 12
        Average precision: 0.9995
        Average recall: 0.9668
        Average F1: 0.9829
        Average accuracy: 0.9998
        Average AUC of ROC: 0.9956


CTU Scenario: 13
        Average precision: 1.0000
        Average recall: 0.9992
        Average F1: 0.9996
        Average accuracy: 1.0000
        Average AUC of ROC: 0.9999
```

In [13]:

```
kfold_precision = np.mean(precision)
kfold_recall = np.mean(recall)
kfold_f1 = np.mean(f1)
kfold_accuracy = np.mean(accuracy)
kfold_auc = np.mean(auc)
```

In [17]:

```
print(f'Average KFold scores for all CTU scenarios:')
print(f'\tAverage precision: {kfold_precision:.4f}')
print(f'\tAverage recall: {kfold_recall:.4f}')
print(f'\tAverage F1: {kfold_f1:.4f}')
print(f'\tAverage accuracy: {kfold_accuracy:.4f}')
print(f'\tAverage AUC of ROC: {kfold_auc:.4f}\n\n')
```

```
Average KFold scores for all CTU scenarios:
        Average precision: 0.9998
        Average recall: 0.9354
        Average F1: 0.9531
        Average accuracy: 0.9999
        Average AUC of ROC: 0.9910
```

Looking at the results for each CTU Scenario, we can see that the average accuracy on the validation is ~99%, which means our model is overfitting the data. This can be combated by increasing the number of folds to 5, and the number of random forest estimators to 100. This would be a much better trained model, however I stuck with the lower number in the interest of time. The F1 score is also averaging around 99%, indicating that the model is able to identify data points in a class properly when validating. The AUC is also close to 1.0, which can mean that the model has a good measure of separability.

NOTE: The initial method of splitting data using numpy was yielding a wider range of numbers, however one dataset was not splitting correctly unless we consolidated all CTU scenarios before running the analysis.

In [ ]:

In [14]:
```python
# Leaves one scenario out and trains on the rest of the scenarios
def leave_one_out(files, scenario_id):
    # Load the feature data from all files except the specified scenario
    features = np.concatenate([load_features(file) for i, file in enumerate(files) if i != scenario_id])
    labels = np.concatenate([np.zeros(features.shape[0] // 2), np.ones(features.shape[0] // 2)])

    # Load the feature data from the specified scenario
    scenario_features = load_features(files[scenario_id])
    scenario_labels = np.concatenate([np.zeros(scenario_features.shape[0] // 2), np.ones(scenario_features.shape[0] // 2)])

    # Define the classifier
    clf = RandomForestClassifier(n_estimators=10, random_state=123)

    # Train the classifier on the rest of the data
    clf.fit(features, labels)

    # Evaluate the classifier on the held-out scenario
    y_pred = clf.predict(scenario_features)
    y_prob = clf.predict_proba(scenario_features)[:, 1]
    precision = precision_score(scenario_labels, y_pred)
    recall = recall_score(scenario_labels, y_pred)
    f1 = f1_score(scenario_labels, y_pred)
    accuracy = accuracy_score(scenario_labels, y_pred)
    auc = roc_auc_score(scenario_labels, y_prob)

    # Return the evaluation metrics
    return precision, recall, f1, accuracy, auc
```

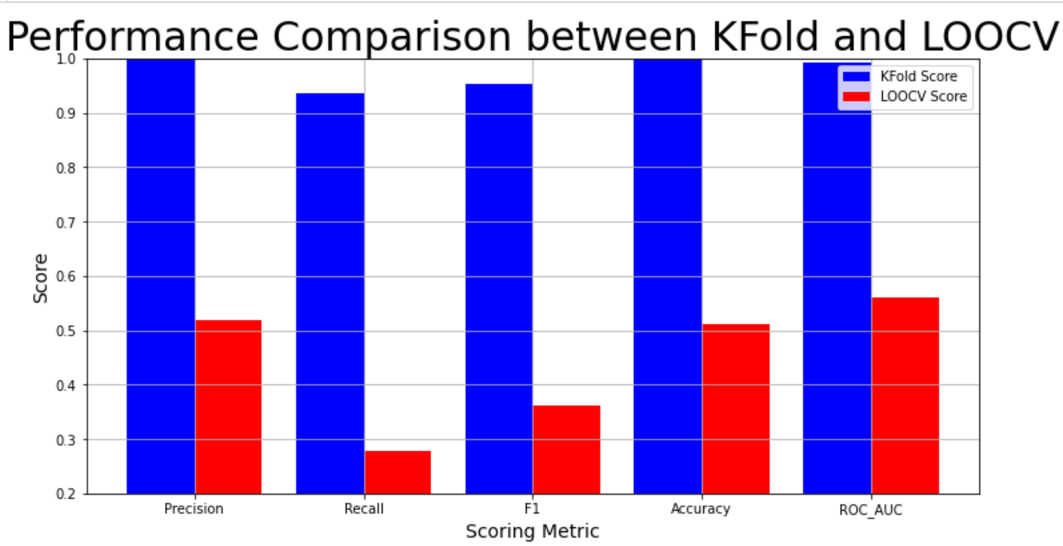
In [15]:
```python
leave_out_index = 0
loocv_precision, loocv_recall, loocv_f1, loocv_accuracy, loocv_auc = leave_one_out(files, leave_out_index)
```

In [18]:
```python
print(f'Average LOOCV scores for all CTU scenarios:')
print(f'\tAverage precision: {loocv_precision:.4f}')
print(f'\tAverage recall: {loocv_recall:.4f}')
print(f'\tAverage F1: {loocv_f1:.4f}')
print(f'\tAverage accuracy: {loocv_accuracy:.4f}')
print(f'\tAverage AUC of ROC: {loocv_auc:.4f}\n\n')
```

```
Average LOOCV scores for all CTU scenarios:
        Average precision: 0.5187
        Average recall: 0.2788
        Average F1: 0.3627
        Average accuracy: 0.5101
        Average AUC of ROC: 0.5604
```

Looking at the scores for the LOOCV model, it seems to be a very inaccurate and underperforming model compared to KFold. This can be seen by the low accuracy and AUC of ROC scores.

In [27]:
```python
plot_result(x_label="Scoring Metric",
            y_label="Score",
            plot_title="Performance Comparison between KFold and LOOCV",
            k_fold_data=[kfold_precision, kfold_recall, kfold_f1, kfold_accuracy, kfold_auc],
            loocv_data=[loocv_precision, loocv_recall, loocv_f1, loocv_accuracy, loocv_auc])
```



This is a visual representation of the performance differences when evaluating the model using KFold vs LOOCV.

This huge performance difference is most likely due to what the two evaluation methods are suited to:

- KFold tends to perform better with larger datasets (like the CTU scenarios) and it can provide a better representation of the underlying data, and be more accurate than LOOCV since it uses multiple splits for training/testing, which in turn reduces the variance in the estimate of a models' performance.
- LOOCV is a more useful evaluation method for smaller datasets, where the higher cost of performance is offset by the lower amount of data. While it does have the advantage of using all of the data in the training set (which reduces bias) compared to the smaller data KFold uses in its splits, the nature of this larger dataset makes LOOCV a very bad performing model for these large datasets.

## References


- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html)
- https://www.section.io/engineering-education/how-to-implement-k-fold-cross-validation (https://www.section.io/engineering-education/how-to-implement-k-fold-cross-validation)
- https://machinelearningmastery.com/loocv-for-evaluating-machine-learning-algorithms/ (https://machinelearningmastery.com/loocv-for-evaluating-machine-learning-algorithms/)
- https://medium.com/analytics-vidhya/step-by-step-guide-to-leave-one-person-out-cross-validation-with-random-forests-in-python-34b2eaefb628 (https://medium.com/analytics-vidhya/step-by-step-guide-to-leave-one-person-out-cross-validation-with-random-forests-in-python-34b2eaefb628)