# CSC 699 Independent Study report 0

By Nicholas Stepanov
Always add date

## Machine Learning methods:

For all classifiers add one major refernce

1. **Random Forest Classifier (RFC)**
   Random Forest Classifier is a Supervised Machine Learning method that utilizes Decision Tree statistical learning. A Decision Tree Classifier constructs a tree that splits on the features of provided dataset to minimize the difference between the actual target class and the output of the tree on every leaf node known as Gini impurity. An advantage of using Random Forest over a Decision Tree method is in the ensemble learning. By returning the mode output of multiple pseudo-randomly created Decision Trees Random Forest Classifier decreases prediction variance compared to a single Decision tree and ensures that there is less overfitting to the training data. Since RFC creates multiple trees it can select features by random having different Decision trees divide the data across various features. This approach makes it possible to better understand importance of features in making a classification decision. The most common parameters to tune a RFC are: the amount decision trees produced, maximum amount of features used in a single Decision Tree and the maxim depth of Decision Trees.

   **How to use with Python SciKit:**

   *from sklearn.ensemble import RandomForestClassifier*

   *x, y = //x - feature set, y - target set*
   *MyClassifier = RandomForestClassifier(random_state=0)*
   *MyClassifier.fit(x,y)*

2. **Support Vector Machines (SVM) with Stochastic Gradient Descent (SGD)**
   Support Vector Machine is a Supervised Machine Learning method which aims to find a hyperplane on the provided dataset that would split the data in classes. Such a hyperplane is chosen to maximize margin - the distance between the closest data points of each class to the hyperplane. The result divides the feature space of dataset into discrete sections that are labeled with the target classes.
   Stochastic Gradient Descent is an optimization technique that can be used for training of an SVM. SGD takes partial derivatives of an error function estimate for a random part of dataset with respect to the dimensions of the feature set and takes small steps defined by learning rate in the direction opposite to the gradient to eventually arrive at the minima of the actual error function. Such a process is repeated until an optimal hyperplane is found. It is important to note that the SGD techniques require normalized data with 0 mean and unit variance to perform best.

**How to use with Python SciKit:**

*from sklearn.linear_model import SGDClassifier*

*x, y = //x - normalized feature set, y - target set*
*MyClassifier = SGDClassifier(loss="hinge")*
*MyClassifier.fit(x,y)*

# Datasets:

Possible datasets for the study:

1.Credit Approval Dataset

Link: https://archive.ics.uci.edu/ml/datasets/Credit+Approval

Comments: 15 features. 690 Instances. The class attribute gives either a positive credit approval or a negative credit approval. Most of the features are encoded to meaningless values which makes it hard to study explainability.I would try this one first. Make sure you process correctly categorical (discrete) vs. continuous features – check how each classifier deals with those

2. Drug consumption based on personality Dataset

Link: https://archive.ics.uci.edu/ml/datasets/Drug+consumption+%28quantified%29

Comments: 12 features + 20 types of drugs that can be used as features or target classes. 1885 instances. More of a regression problem - the drugs are marked with use intensity, but could be reduced to a classification problem by taking values above certain point as positive and other as negative. The feature set is very meaningful. Skip this one

3.Epileptic Seizure Recognition based on Brain activity Dataset

Link: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition

Comments: 179 features, 5 target classes, but only one target class has epileptic seizures. 11500 instances The features are meaningful EEG scan values, but are hard to interpret by a non-expert. skip

4. Grammatical Facial Expressions Dataset

Link: https://archive.ics.uci.edu/ml/datasets/Grammatical+Facial+Expressions

Comments: 100 features, 18 classes. 27965 instances. Large amount of classes, but could be reduced to a single class for a specific expression. The features are the positions of facial points recorded through Kinect and are not very easy to interpret.

5. Wine quality based on Chemical characteristics Dataset

Link: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Comments: 11 features + a quality score. 4898 Instances. More of a regression problem, but could be reduced to classification like good wine/bad wine. The features are pretty intuitive chemical properties, but not very easy to interpret for a non-expert.

6. Software Engineering Teamwork Assessment Database

Link: https://archive.ics.uci.edu/ml/datasets/Data+for+Software+Engineering+Teamwork+Assessment+in+Education+Setting

Comments: 100 features. 74 Instances. The features are intuitive to interpret. Small amount of instances compared to other datasets. Skip this one, we did it before. See attcahed paper

# F1 score accuracy:

**1.a) F1 score accuracy metrics**: weighted average of precision and recall

**F1 score** = 2*(**Recall***Precision**)/(**Recall** + **Precision**) , where

**1.b) Recall**: The ratio of True Positive predictions in all the data marked as Positive

**Recall** = True Positive#/(True Positive# + False Negative#)

**1.c) Precision**: The ratio of True Positive predictions in all Positive predictions

**Precision** = True Positive#/(True Positive# + False Positive#)

**2.To see True/False Positives/Negatives table use**:

*pandas.crosstab(y_test, MyClassifier.predict(X_test), rownames=['Actual'], colnames=['Predicted'])*

**3.To see F1 score use:**

*from sklearn.metrics import f1_score*

*Y_true = y_test*
*Y_pred = MyClassifier.predict(X_test)*
*f1_score(Y_true, Y_pred, average="weighted")*

Important thing for F1 score computation is to vary "sensitivity" of classifiers. In RF you vary CTOFF which is the threshold used to determine fraction of the trees in RF forest which needs to vote for winnign class. (Defauilt in RF is 50%). Check it out. So, we optimize RF by varying ntree and mtry, and for each fo those combinations vary cutooff 10%, 20%...90% and check which of those produces highest F1 score.

For SVM do reserch how to vary sensitivity to get F1 score. I forgot but you can read this paper of ours – we used AUC or Area under the Curve there as accuracy measure https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0091240