Module: Machine Learning

Live Session-1

Agenda:
Basic introduction to ML
ML: Evaluation metrics
scikit-learn library
Naïve Bayes Classifier

About the Module:

- Regression Models (concepts; Demo; Implementation; etc.)
- Support Vector Machines (SVM; SVR; Advanced level implementation)
- K Nearest Nhbr.; K-Means Clustering; Hierarchal Clustering
- Decision Trees; C4.5; CART etc.
- Some more classifiers and Regressions

Machine Learning: Categorization

- **Supervised machine learning algorithms** can apply what has been learned in the past to new data using labeled examples to predict future events.
- In contrast, **unsupervised machine learning algorithms** are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data.
- **Semi-supervised machine learning algorithms** fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data.
- **Reinforcement machine learning algorithms** is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning.

Machine Learning: Classification

- Classification is a part of **supervised learning** (learning with labeled data) through which data inputs can be easily separated into categories. There can be **binary classifiers** or **multi-class classifiers**.

- Some of the classification applications are face recognition, Youtube video categorization, content moderation, medical diagnosis, to text classification, hate speech detection on Twitter.

- Some classification models are given as support vector machine (SVM), logistic regression, decision trees, random forest, convolutional neural network, recurrent neural network.

Performance Evaluation: Confusion Matrix

- One of the Key Concept in Classification performance is **Confusion Matrix** which is a tabular visualization of the model predictions versus the actual labels.
- Each row of the confusion matrix represents the instances in an actual class whereas each column represents the instances in a predicted class.
- Consider a binary classification problem having Cat and Non-Cat images. Just assume that Cat class as positive class. Then,

		Predicted Class	
		Cat	Non-cat
Actual Class	Cat	90	10
	Non-Cat	60	940

TP=90; TN=940; FP=60; FN=10

https://colab.research.google.com/drive/1qpwsj3U7T31f3qCRcgAXEtDQCXIWqqQw

Classification accuracy

The simplest metrics of classification accuracy is defined as the **number of correct predictions divided by the total number of predictions,** multiplied by 100.

Classification-accuracy

$$=(90+940)/(1000+100)$$

$$=1030/1100$$

$$= 93.6\%$$

Precision

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances.

Precision= True_Positive/(True_Positive+ False_Positive)

Precision_cat= #samples correctly predicted cat/#samples predicted as cat = 90/(90+60) = 60%

Precision_NonCat= 940/950= 98.9%

As we can see the model has much higher precision in predicting non-cat samples, versus cats. This is not surprising, as model has seen more examples of non-cat images during training, making it better in classifying that class.

Recall

Recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved.

Recall= True_Positive/ (True_Positive+ False_Negative)

Recall_cat= 90/100= 90%

Recall_NonCat= 940/1000= 94%

F1-score

Harmonic mean of precision and recall (weighted average)

F1-score= 2*Precision*Recall/(Precision+Recall)

F1_cat= 2*0.6*0.9/(0.6+0.9)= 72%

It is good to mention that there is always a trade-off between precision and recall of a model, if you want to make the precision too high, you would end up seeing a drop in the recall rate, and vice versa.

https://colab.research.google.com/drive/1qpwsj3U7T31f3qCRcgAXEtDQCXIWqqQw

ROC and AUC

The **Receiver Operator Characteristic (ROC)** curve is an evaluation metric for binary classification problems. It is a probability curve that plots the **TPR** against **FPR** at various threshold values and essentially **separates the 'signal' from the 'noise'**.

The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

https://colab.research.google.com/drive/1AFdwmMmKnO6PLfZ_K6z91vKkfnPUAmG2#scrollTo=RGAwo9LV4F92&uniqifier=1

Scikit-Learn or Sklearn

Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning.

It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

- Fitting and predicting: estimator basics
- Transformers and pre-processors
- Pipelines: combining pre-processors and estimators
- Model Evaluation
- Automatic parameter searches

Sklearn: Fitting and Predicting

- Scikit-learn provides dozens of built-in machine learning algorithms and models, called estimators.
- Each estimator can be fitted to some data using its fit method.
- Example:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=0)
X = [[ 1,  2,  3],  # 2 samples, 3 features
...      [11, 12, 13]]
y = [0, 1]  # classes of each sample
clf.fit(X, y)
clf.predict([4,5,6],[11, 11, 13])
```

https://colab.research.google.com/drive/1SRsDMJQpNgbk30F3FUxJtyCy4PuH_qva#scrollTo=nQ4OxcxqY8cm

fit and predict methods

The fit method generally accepts 2 inputs:

- The samples matrix (or design matrix) X. The size of X is typically (n_samples, n_features), which means that samples are represented as rows and features are represented as columns.
- The target values y which are real numbers for regression tasks, or integers for classification (or any other discrete set of values). For unsupervized learning tasks, y does not need to be specified. y is usually 1d array where the i th entry corresponds to the target of the i th sample (row) of X.

Both X and y are usually expected to be numpy arrays or equivalent array-like data types, though some estimators work with other formats such as sparse matrices.

Once the estimator is fitted, it can be used for predicting target values of new data.

Pre-processors

- Machine learning workflows are often composed of different parts.
- A typical pipeline consists of a pre-processing step that transforms or imputes the data, and a final predictor that predicts target values.
- The transformer objects don't have a predict method but rather a transform method that outputs a newly transformed sample matrix *X*:

```
from sklearn.preprocessing import StandardScaler
X = [[0, 15],
...      [1, -10]]
# scale data according to computed scaling values
StandardScaler().fit(X).transform(X)
```

Processed data follows Gaussian distribution with 0 mean and unit variance

https://colab.research.google.com/drive/1_YVTkcfqXRQOn5EGmMDbmpgMS8JLTLoK

Pipelines: chaining pre-processors and estimators

Preprocessors and estimators (predictors) can be combined together into a single unifying object: a Pipeline.

The pipeline offers the same API as a regular estimator: it can be fitted and used for prediction with fit and predict.

https://colab.research.google.com/drive/1r3fc5t1A0VoZEEVx7JByPXi52r4AZDU?usp=sharing

https://colab.research.google.com/drive/18CVT3M0Z7adt20GfcHDTyWa4WTs7B4pE#scrollTo=hXZf-9gm5OdT

Model evaluation

- Fitting a model to some data does not entail that it will predict well on unseen data.
- This needs to be directly evaluated. We have just seen the train_test_split helper that splits a dataset into train and test sets, but scikit-learn provides many other tools for model evaluation, in particular for cross-validation.
- We here briefly show how to perform a 5-fold cross-validation procedure, using the cross_validate helper.
- Note that it is also possible to manually iterate over the folds, use different data splitting strategies, and use custom scoring functions.

https://colab.research.google.com/drive/1hGKLPtQ43Le6xjsH9qp7bCt1JClYh9a4?usp=sharing

Automatic Parameter Search

- All estimators have parameters (often called hyper-parameters in the literature) that can be tuned.
- The generalization power of an estimator often critically depends on a few parameters. For example a RandomForestRegressor has a n_estimators parameter that determines the number of trees in the forest, and a max_depth parameter that determines the maximum depth of each tree.
- Quite often, it is not clear what the exact values of these parameters should be since they depend on the data at hand.

Scikit-learn provides tools to automatically find the best parameter combinations (via cross-validation).

https://colab.research.google.com/drive/1J9XL82kqMItqmdVxpnuMA1Ez6p4dLqJs

Naïve Bayes Classifier

- Let's say that we are interested in knowing whether an e-mail that contains the word *freeloan* (event) is spam (hypothesis). If we use the Bayes theorem description, this problem can be formulated as:

$$P(class = Spam | Contains = freeloan)$$

$$= \frac{P(Contains = freeloan | Class = Spam) * P(class = spam)}{P(contains = freeloan)}$$

Naïve Bayes Classifier

P(class=SPAM|contains="freeloan") is the probability of an e-mail being SPAM given that this e-mail contains the word freeloan. This is what we are interested in predicting.

P(contains="freeloan"|class=SPAM) is the probability of an e-mail containing the word freeloan given that this e-mail has been recognized as SPAM. This is our training data, which represents the correlation between an e-mail being considered SPAM and such e-mail containing the word freeloan.

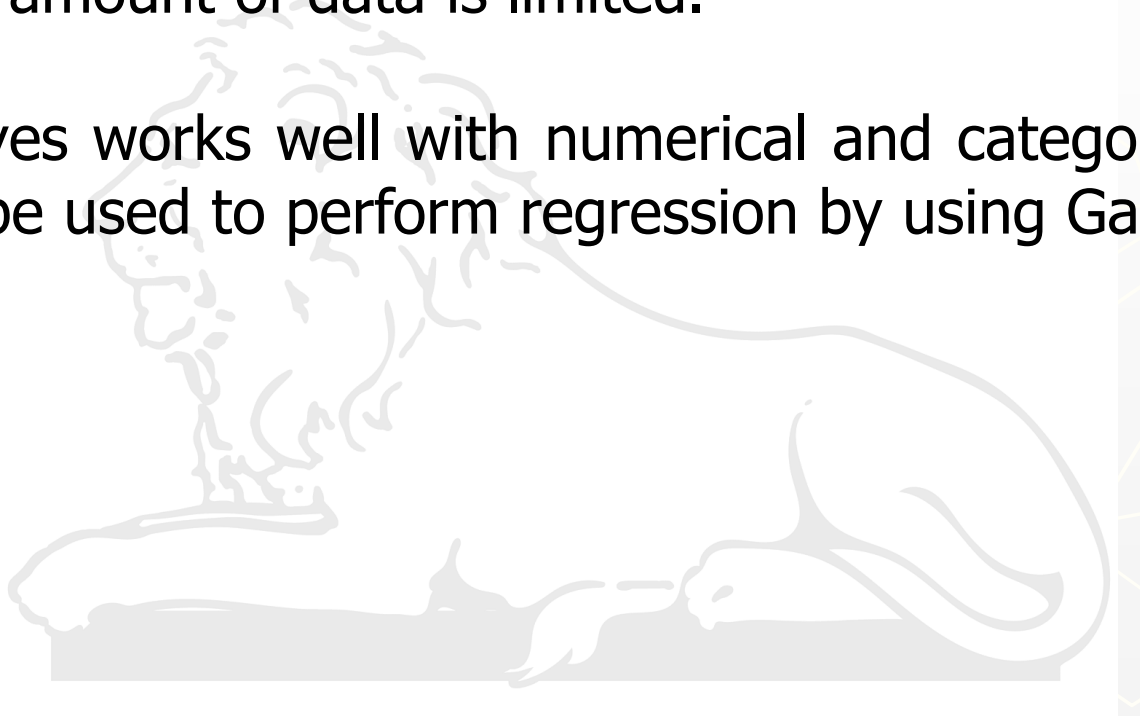
P(class=SPAM) is the probability of an e-mail being SPAM (without any prior knowledge of the words it contains). This is simply the proportion of e-mails being SPAM in our entire training set. We multiply by this value because we are interested in knowing how significant is information concerning SPAM e-mails. If this value is low, the significance of any events related to SPAM e-mails will also be low.

P(contains="freeloan") is the probability of an e-mail containing the word freeloan. This is simply the proportion of e-mails containing the word freeloan in the entire training set. We divide by this value because the more exclusive the word freeloan is, the more important is the context in which it appears.

Naïve Bayes: Pros

- Naive Bayes is a simple and easy to implement algorithm. Because of this, it might outperform more complex models when the amount of data is limited.

- Naive Bayes works well with numerical and categorical data. It can also be used to perform regression by using Gaussian Naive Bayes.



Naïve Bayes: Cons

Given the construction of the theorem, it does not work well when you are missing certain combination of values in your training data.

In other words, if you have no occurrences of a class label and a certain attribute value together (e.g. class="spam", contains="$$$") then the frequency-based probability estimate will be zero. Given Naive-Bayes' conditional independence assumption, when all the probabilities are multiplied you will get zero.

THANK YOU