Program reskilling Data/AI PO
Specific session for SOCIETE GENERALE

3: Supervised learning logistic regression decision trees and neural network

Eric Benhamou

PSL UNIVERSITÉ PARIS Dauphine UNIVERSITÉ PARIS ENS MINES ParisTech

PR[AI]RIE PaRis Artificial Intelligence Research InstitutE

Agenda

Session 3: Classification

The various classification models

- Logistic regression model
- Decision trees
- Recursive features elimination

Metrics of model performance

- ROC Curve
- Confusion matrix
- Classification reports

LAB

- Building the model
- Choose features with RFE
- Train models (baseline model with logistic regression and Decision trees)
- Compare with a deep network
- Visualize models performance
- Save model for production

Summary

Recall from previous session

- In the previous session, you learned about **time series analysis**, the benefits of time series analysis for banking and **two useful examples** by defining the problem statement and **deriving the solution step by step**.

- We also learned about the basic concepts of time series analysis and a few techniques, such as **ARIMA** and **deep learning**

Goal of this session

- Commercial banks make money by earning interest on money that was loaned to borrowers. In many cases, the loan becomes a Non-Performing Asset (NPA) for the bank. There are instances where the borrower could go bankrupt, leaving the bank with a loss. In such situations, it becomes critical for commercial banks to assess the borrower's ability to repay the loan in a timely manner.

- In this session, we will learn about different AI modelling techniques to do classification with an example for predicting the chances of the borrower going bankrupt. The algorithms that we will investigate are
  - logistic regression model,
  - decision trees,
  - and deep learning.

- We will also learn about the various metrics of model performance to validate if a model has learnt something and how to pick the best model

Motivations

- Within a bank, as an intermediary between those with excess money (the depositors) and those who need money (the borrowers), there are two important questions that need to be answered:
  - How risky is a borrower?
  - What is the funding cost of money?

- These are the two important questions that need to be considered before we look at the profit required for sustaining the business operations in order to cover its running costs.

- When these decisions are not made properly, it threatens the viability of a bank. There could be two possible outcomes in such instances:
  - If the bank does not make enough profit to cover the cost of risk and operations when a risky event occurs, the bank could collapse.
  - If the bank fails to meet the depositor's requirements or fails to honor its borrower's agreements to lend, it hurts the credibility of the bank, thus driving potential customers away.

Major types of risk

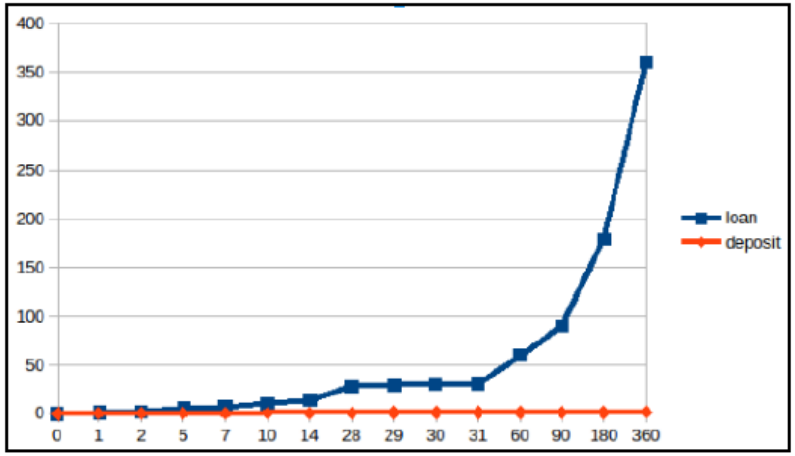
- To answer the question, *How risky is a borrower?,* we first need to understand the factors contributing to risk.
- Risk is an unfavorable outcome in the future that impacts the functioning of a bank. For a bank, the major contributors include the following:
  - Credit risk: This risk concerns the borrower's inability to repay the capital back to the bank in a lending transaction; for example, the financial distress of the borrowing firm, causing its inability to repay the loan.
  - Market risk: This risk concerns unfavorable price movements in financial markets, such as an interest rate hike in the market from which the bank sources its funding.
  - Operational risk: This risk concerns events happening in the operations of the bank as an organization. This could include internal theft, a cyber attack, and so on.
- For a complete list of the types of risk, please refer to the Basel Framework by BIS (https://www.bis.org/bcbs/basel3.htm).

Asset liability management

- Commercial banks need deposits in order to fund loans. As well as assessing the riskiness of borrowers, the bank also performs a useful function in that they convert deposits from savers into loans for borrowers. Thus, a pricing mechanism for both depositors and borrowers is important. To a bank, loans sit on the asset side of financial statements, while deposits sit on the liabilities side of the business. Therefore, this is often called Asset and Liability Management (ALM).
- Ignoring other risks such as liquidity risk, interest rate risk, and foreign exchange risk, the objectives of the ALM function of a bank are the following:
  - Ensure loans are supported by deposits and that the bank will have sufficient deposits, in case the depositors ask for their money back. In terms of the total quantity, approximately, a $100 deposit supports a $70 loan. Referencing the ratios from some of the biggest banks, the ratios should be around 1.2:1 to 1.5:1 for a customer deposit to a customer loan.
  - Ensure over time, the loan is viable, which is measured by concept of the duration. To meet long-term loan commitments, the bank also needs deposits to be locked in for a long enough time to ensure that loans are supported by deposits in a long- term manner.
  - Thirdly, ensure ALM is profitable, which means the ALM income should be higher than the ALM cost. The *cost* is the ALM pricing that you are giving out. This cost is, in fact, the income for ALMs/banks, while the deposit rate quoted to the client is the bank's expense.

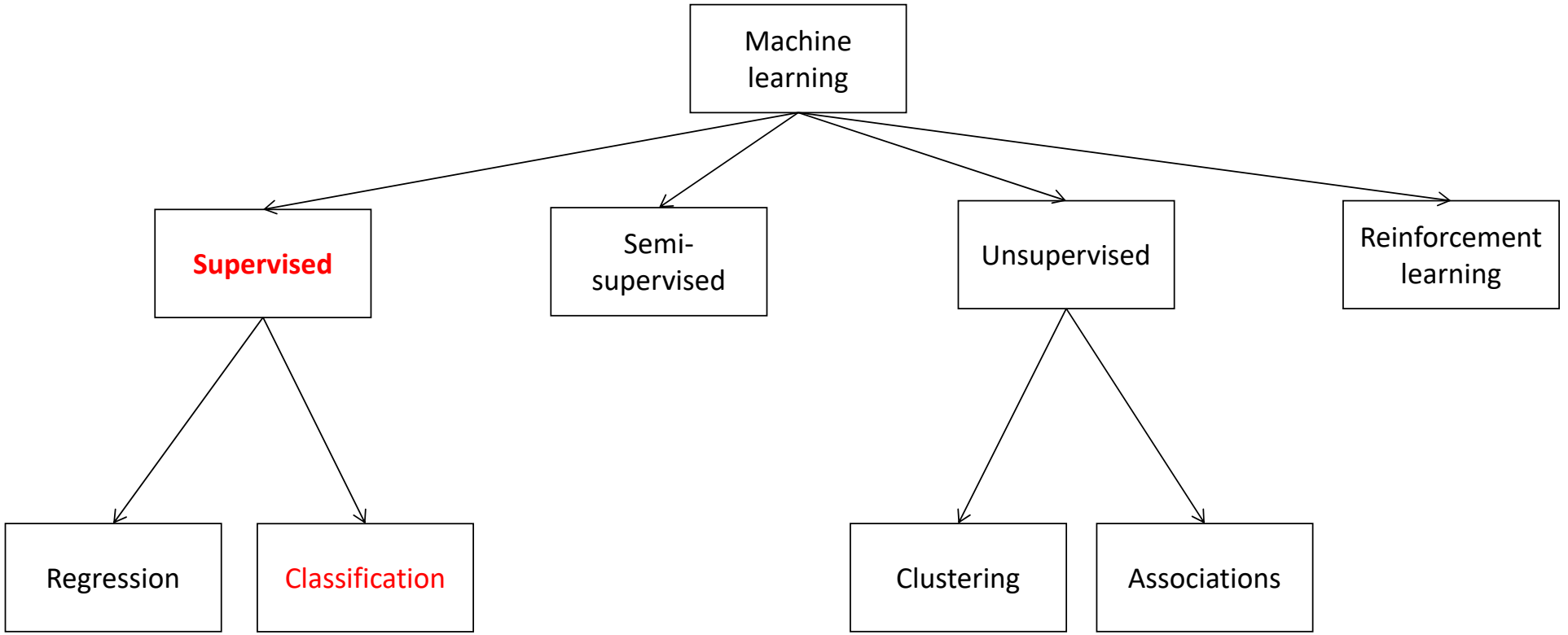
All is about conversion

- Part of a bank's well-known secret for profit is to convert the short-term deposit (lower priced) into a long-term loan (higher interest income). The following curve shows the pricing aspect for a bank for its deposits and loans:



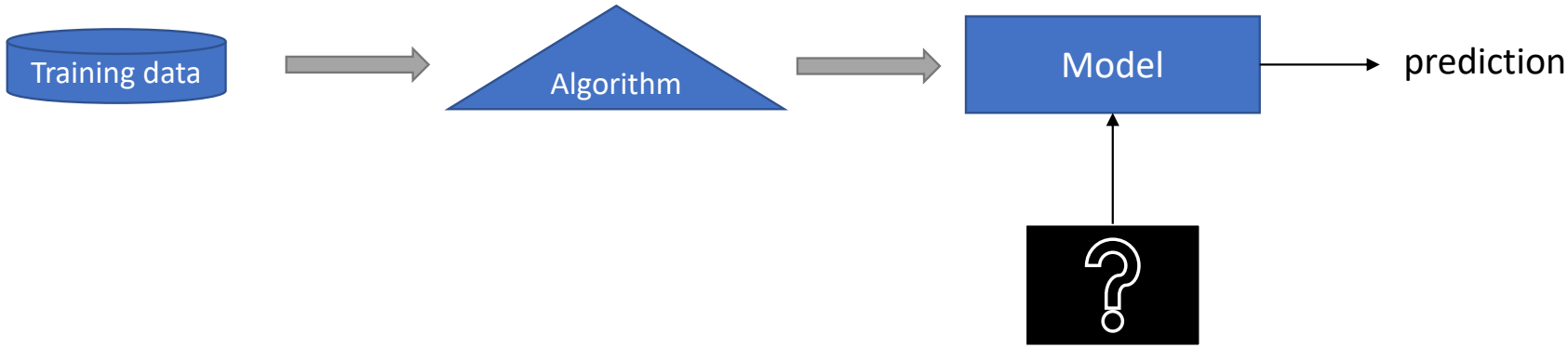
- In the preceding graph, the x axis shows how long (in days) the deposit/loan position will remain with the bank, while the y axis shows the annualized interest rate.

ML taxonomy


Machine learning
Supervised
Semi-supervised
Unsupervised
Reinforcement learning
Regression
Classification
Clustering
Associations

Supervised learning

- In supervised learning, the algorithms are presented with a set of classified instances from which they learn a way of classifying unseen instances. When the attribute to be predicted is numeric rather than nominal it is called regression.



Logistic regression model

- The logistic regression model is **one of the most popular adoptions** of AI in banking, especially in the **domain of credit risk modeling**. The target variable of the model will be a binary outcome of 1 or 0, with a probability of meeting the target of 1. The decision of what 1 and 0 refer to depends on how we prepare the data.

- As an optimization problem, binary class ℓ2 penalized logistic regression minimizes the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

- The model is called logistic because the function that models the 1 and 0 is called **logit**. It is called **regression** because it belongs to a statistical model called the regression model, which strives to determine the causation of factors of an outcome

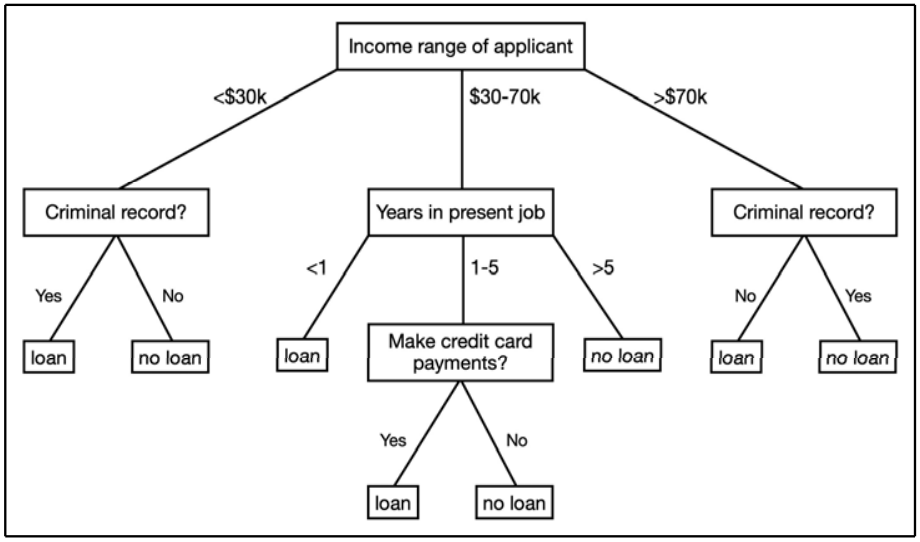
implementation

- Use sklearn.linear_model.LogisticRegression
- Choose the solver

	Solvers				
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Elastic-Net	no	no	no	no	yes
No penalty ('none')	no	yes	yes	yes	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

Decision trees

- The decision tree algorithm actually belongs to the supervised learning group of algorithms. However, due to the nature of the algorithm, it is commonly used to solve regression and classification problems. Regression and classification often require decision- making based on the situation at hand. So, these problems are commonly solved using reinforcement learning algorithms.
- The beneficial element of having a decision tree is that we can actually visualize the decision tree's representation. The decision-making process starts at the top of the tree and branches out toward the leaf nodes of the tree. The leaf nodes are the point at which the target variables will end up. All the values of a variable that are classified to the same leaf node contain the same probability of defaulting. The following is an example visualization of a decision tree algorithm that is making a decision to give a loan to the applicant



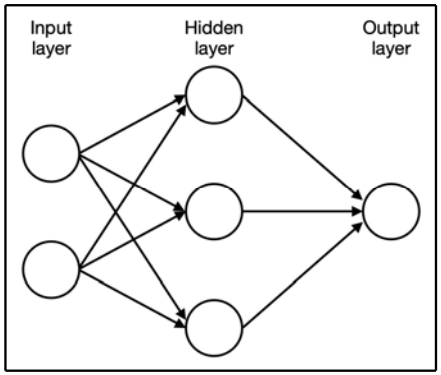
Decision trees: interpretability

- The most common way to move forward in the decision tree is to look at the minimal leaf size, which refers to the size of the bucket that each of the training samples is being classified in. If the bucket contains too few samples than min_samples_leaf dictates, then it will be scrapped. This can be done to reduce the number of buckets (known as the leaf node of a decision tree).

- Reading the decision tree is easy. However, it is quite amazing to realize how the machine learns about the various conditions used for splitting

Neural networks 1/2

- A simple neural network looks like the one shown in the following diagram:



- It consists of three layers, namely the input layer, the hidden layer, and the output layer. Each layer is made up of nodes. The artificial neural network that is used to solve AI problems mimics the physical neural network present in the human brain. The neurons in the human brain are represented by nodes in the artificial neural network. The connections between the neurons are represented in the artificial neural network by weights.
- Let's understand the significance of each of the layers in the neural network. The input layer is used to feed the input into the model. It is also responsible for presenting the condition that the model is being trained for. Every neuron or node in the input layer represents one independent variable that has influence over the output.

Neural networks 2/2

- The hidden layer is the most crucial because its job is to process the data it has received from the input layer and is responsible for extracting the necessary features from the input data. The hidden layer consists of one or more layers.

- In the case of solving a problem with linearly represented data, the activation function (which processes the input data) can be included in the input layer itself. However, for processing complex representations of data, one or more hidden layers are required and non linear activation is requireds. The number of hidden layers depends on the complexity of the data. The hidden layer passes on the processed data to the output layer.

- The output layer is responsible for collecting and transmitting information. The pattern that the output layer presents can be traced back to the input layer. The number of nodes in the output layer depends on the number of decisions to be made eventually

Metrics of model performance

- When we build an AI model, the most important aspect of the process is to define a way to measure the performance of a model. This enables the data scientist to decide how to improve and pick the best model.
- In this section, we will learn about three common metrics that are commonly used in the industry to assess the performance of the AI model.
  - ROC curve
  - **confusion matrix**
  - **classification report**

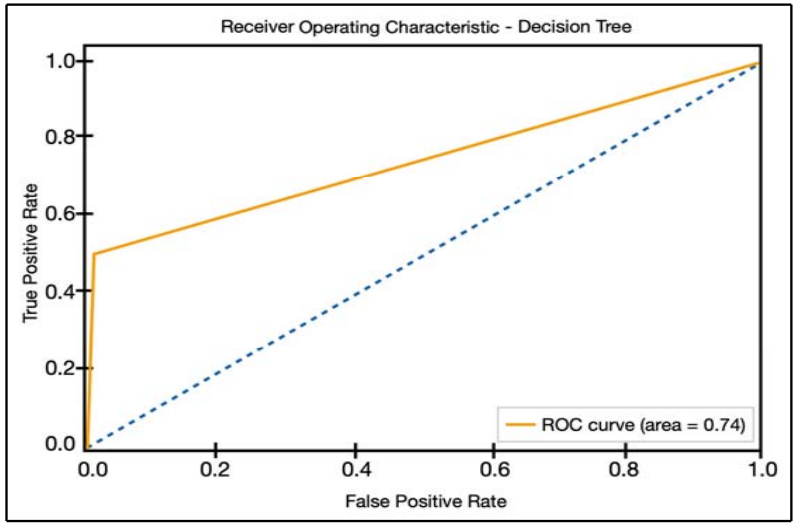
Metric 1: ROC curve 1/3

- The Receiver Operating Characteristic (ROC) metric measures how well the classifier performs its classification job versus a randomized classifier. The classifier that's used in this metric is a binary classifier. The binary classifier classifies the given set of data into two groups on the basis of a predefined classification rule.
- This is linked to a situation where, say, we compare this model against flipping a fair coin to classify the company as being default or non-default, with heads indicating default and tails indicating non-default. Here, there's a 50% chance of classifying default and a 50% chance of classifying non-default.

- For a completely randomized predictive system such as coin flipping, it is very likely that the probability of hitting a true positive is the same as hitting a false positive rate. But in the case of companies defaulting in 1 year, in the following example, it is 6.3% (123 out of 1,828), which means we have an actual count of 1,828 non-default cases and 123 default cases. A truly random model will predict half of the default cases as non-default.

Metric 1: ROC curve 2/3

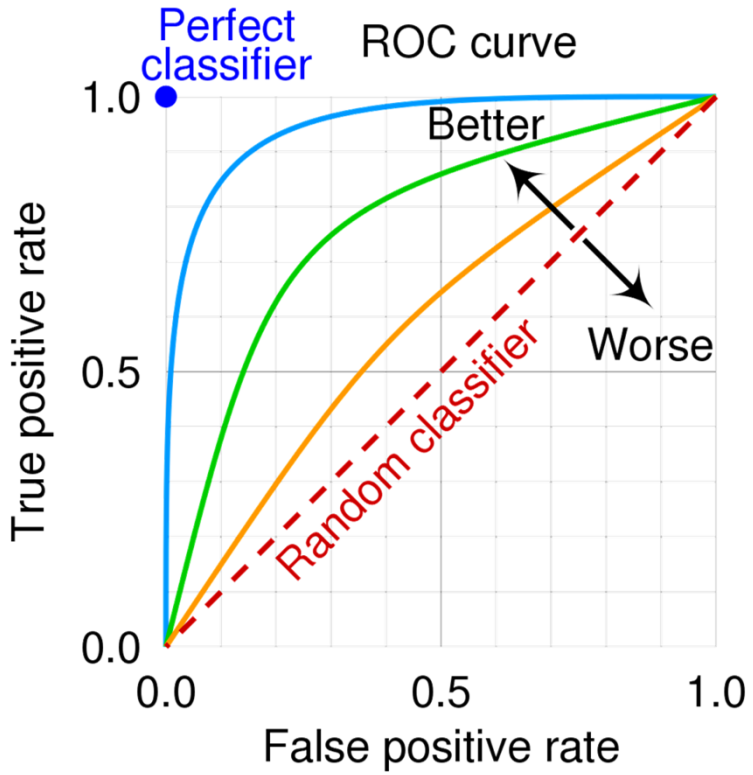
- Let's plot a chart that shows the true positive and false positive rate as an ROC chart. True or false means the prediction that was made for the default event is factually true or false. Positive means that the classifier is positive (equals 1, which is default, in this case).
- When we make no prediction, the true positive and false positive rate is 0. When we have gone through 50% of the sample, which is given as 1,951/2, we should be getting 50% of the sample by default, where 50% of the guesses are false positive. When we get to 100% of the sample, we should have 100% of the sample as true positive and 100% as false positive.
- This randomized classifier's performance is denoted by the dotted line in this diagram:



Metric 1: ROC curve 3/3

- In the most ideal classifier case, we should be able to improve the true positive rate to 100%, with the false positive rate at 0% (denoted by the yellow line in the preceding diagram).

- For the worst classifier, which classifies everything as 100% incorrect, the true positive rate should be 0% and the false positive rate should be 100% (denoted by the red dot). The use of ROC is also prevalent in credit risk model validation.



Metric 2: Confusion matrix

- The confusion matrix is the most popular metric used to measure the performance of a classifier and has two outcomes:

		Actual: Ground Truth	
		True Default	False/Non-default
Prediction by Classifier	Positive/Default	62	27
	Negative/Non-default	61	1,801
		True Positive Rate = 62/(62+61)	False Positive Rate = 27/(27+1,801)

- The confusion matrix also provides results similar to the ROC curve. The major idea behind this is to separate prediction and the ground truth by rows and columns.

Metric 3: Classification report 1/2

- The classification report is another way to appraise the performance of the model, with the following indicators:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	1828
1	0.69	0.50	0.58	123
avg / total	0.95	0.95	0.95	1951

Metric 3: Classification report 2/2

- The details of the indicators are as follows:
  - Precision and recall: Precision addresses the true positive rate of the model prediction, while recall addresses the coverage of the model. Precision measures the percentage of the predicted value being the predicted value. Recall measures the percentage of the target values being predicted as the expected values.
  - F1-score: One of the most important measures of the overall accuracy of the model is the F1-score. It is the harmonic mean of precision and recall. This is what we use to compare the performance of models.
  - Support: This is another term that means the number of records that are of the value listed in the leftmost column. There are 123 actual default cases (with target *value = 1* under the *default* column).

Code: import all the relevant libraries

```
'''**********************************
1. Import libraries and define key variables
'''
import os
import re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import classification_report,roc_curve, auc,confusion_matrix,f1_score
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE
from sklearn import linear_model,tree
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler

import pickle
import graphviz
```

Need for RFE (Recursive Features Elimination)

For logistic regression, when it comes to deciding which features are to be chosen, we will rely on testing the accuracy of different features. The combination that delivers the highest accuracy will be chosen.

Define the optimize_RFE() function, which will perform the feature selection process. This function will try out different combinations of features that give the highest true positive and the lowest possible false positive. We will measure the performance in order to decide on the number of features that generate the best performance. We will shortly define the code for that purpose:

Code: define variables

- '''**********
  1. Define program-wide variables and values
  '''

```
#set up the working directory where datafiles are all located
data_path = os.getcwd()
os.chdir(data_path)

#read in files
file_attrib_in = os.path.join(data_path,'attrib.txt')
file_path_in = os.path.join(data_path,'5year.csv')
file_path_out = os.path.join(data_path,'output_dataset.txt')
file_corr_out = os.path.join(data_path,'corr.txt')
f_name = pd.read_csv(file_path_in,sep=',')
f_attrib = open(file_attrib_in,"r")
attrib_str = f_attrib.read()
```

Code: defined more variables

```
#assign column headers
label_name = 'default'
re_obj = re.compile(r'X[0-9]+\s')
fields_list = re_obj.split(attrib_str)
fields_list = fields_list[1:]
fields_list.append(label_name)
f_name.columns = fields_list

#create X and Y dataset with the right header
X = f_name.iloc[:,:-1]
Y= f_name.iloc[:,-1]
#make sure data types are correct and missing values are handled
Y=Y.astype(int)
cols = X.columns[X.dtypes.eq(object)]
for c in cols:
    X[c] = pd.to_numeric(X[c], errors='coerce')
X=X.fillna(0)
```

Code: Define function helpers

- '''
  2. Define all the functions
  '''
  '''
  ##2A. Logistic Regression model
  '''

```
##2A.i
#input the dataframe and the list of columns wanted from it, it return the dataframe with columns selected
def select_columns(df, col_list):
    df_selected = df[df.columns.intersection(col_list)]
    return df_selected

##2A.ii
#input the support (true/false) of the column lists, and the column header, return the list only with true value
def generate_column_lists(col_support,col_list):
    i = 0
    select_cols = []
    len_list = len(col_list)
    while i< len_list:
        if col_support[i]:
            select_cols.append(col_list[i])
        i=i+1
    return select_cols
```

Code: define RFE 1/2

```
##2A.iii
##try any number of features, return the #of features that deliver the best accuracy (AUC)
def optimize_RFE(logreg, X, Y, min_features_to_select = 5):
    trial_cnt = 1
    max_roc_auc=0
    #best_feature = 0
    best_col_list = []
    result_list = {}
    col_list = list(X.columns.values)

    rfe = RFE(logreg,verbose=1, min_features_to_select=min_features_to_select)
    rfe = rfe.fit(X,Y)
    print(rfe.support_)
    print(rfe.ranking_)
    col_support = rfe.support_

    #select the columns
    select_cols = generate_column_lists(col_support, col_list)

    #generate the dataframe with only the list of columns
    X_selected = select_columns(X,select_cols)
    print(list(X_selected.columns))

    #build model
    print('split data')
    X_train, X_test, Y_train, Y_test = train_test_split(X_selected, Y, test_size=0.33, random_state=42)
    print('build model')
    logreg.fit(X_train,Y_train)
    Y_score = logreg.decision_function(X_test)
```

Code: define RFE 2/2

```
    ##metric 1: roc
    fpr, tpr, thresholds = roc_curve(Y_test,Y_score, pos_label=1)
    roc_auc = auc(fpr,tpr)

    result_list[trial_cnt] = roc_auc
    result_list['F_'+str(trial_cnt)] = select_cols

    #memorize this setting if this ROC is the highest
    if roc_auc > max_roc_auc:
        max_roc_auc = roc_auc
        #best_feature = trial_cnt
        best_col_list = select_cols
        print('roc_updated at '+ str(trial_cnt))


    return max_roc_auc, best_col_list, result_list
```

Code: train the model 1/3

```
##2A.iv
#feed in data to the logistic regression model
def train_logreg(X,Y):
    print('Logistic Regression')
    logreg = linear_model.LogisticRegression(C=1e5, solver='saga')
    roc_auc, best_col_list, result_list = optimize_RFE(logreg, X,Y)

    #split the dataset into training set and testing set
    X_selected = select_columns(X, best_col_list)
    X_train, X_test, Y_train, Y_test = train_test_split(X_selected, Y, test_size=0.33, random_state=42)

    #fit the training data to the model
    #preprocessing the data
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)
    logreg.fit(X_train,Y_train)

    ##metric 1: roc
    Y_score_logreg = logreg.decision_function(X_test)
    fpr, tpr, thresholds = roc_curve(Y_test,Y_score_logreg, pos_label=1)
    roc_auc = auc(fpr,tpr)
    lw=2
    plt.figure()
    plt.plot(fpr,tpr,color='darkorange',lw=lw,label='ROC curve (area = %0.2f)' %roc_auc)
    plt.plot([0,1],[0,1],color='navy',lw=lw,linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic - Logistics Regression Model')
    plt.legend(loc="lower right")
    plt.show()
```

Code: train the model 2/3

```
##2A.iv
#feed in data to the logistic regression model
def train_logreg(X,Y):
    ....

    ##metric 2: Confusion matrix
    Y_pred_logreg = logreg.predict(X_test)
    confusion_matrix_logreg = confusion_matrix(Y_test, Y_pred_logreg)
    print(confusion_matrix_logreg)
    print(classification_report(Y_test, Y_pred_logreg))

    #common standard to compare across models
    f1_clf = f1_score(Y_test, Y_pred_logreg, average='binary')

    ##Quality Check: tets for dependency
    corr_m = X_selected.corr()
    sns.heatmap(corr_m)
    corr_m.to_csv(file_corr_out)
    plt.show()
```

Code: train the model 3/3

```
##2A.iv
#feed in data to the logistic regression model
def train_logreg(X,Y):
    ....

    ##3. save model
    f_logreg=open('log_reg.pkl',"wb+")
    pickle.dump(logreg, f_logreg)
    f_logreg.close()

    f_logreg_sc = open('logreg_scaler.pkl',"wb+")
    pickle.dump(scaler, f_logreg_sc)
    f_logreg_sc.close()

    print('These columns are in the final model')
    print(best_col_list)
    thefile = open('logreg_cols.txt', 'w+')
    for item in best_col_list:
        thefile.write("%s\n" % item)
    '''
    [[1790   21]
    [ 118   22]]
            precision    recall  f1-score   support

         0       0.94      0.99      0.96      1811
         1       0.51      0.16      0.24       140

    avg / total       0.91      0.93      0.91      1951
    '''
    return logreg, f1_clf
```

Code: decision trees 1/2

```
'''
##2B. Decision Tree
'''
##2B.i
#feed in data to the decision tree
def train_tree(X,Y):
    print('Decision Tree')
    #split the dataset into training set and testing set
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.33, random_state=0)

    min_leaf_size = int(len(X_train) * 0.01)
    tree_clf = tree.DecisionTreeClassifier(min_samples_leaf=min_leaf_size)

    #preprocessing the data
    scaler = StandardScaler()
    scaler.fit(X_train)

    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    #fit the training data to the model
    tree_clf.fit(X_train,Y_train)

    ##metric 1: roc
    Y_score_tree = tree_clf.predict(X_test)
    fpr, tpr, thresholds = roc_curve(Y_test,Y_score_tree, pos_label=1)
    roc_auc = auc(fpr,tpr)
    lw=2
    plt.figure()
    plt.plot(fpr,tpr,color='darkorange',lw=lw,label='ROC curve (area = %0.2f)' %roc_auc)
    plt.plot([0,1],[0,1],color='navy',lw=lw,linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic - Decision Tree')
    plt.legend(loc="lower right")
    plt.show()
```

Code: decision trees 2/2

```
'''
##2B. Decision Tree
'''
##2B.i
#feed in data to the decision tree
def train_tree(X,Y):
    …


    ##metric 2: Confusion matrix
    Y_pred_tree = tree_clf.predict(X_test)
    confusion_matrix_tree = confusion_matrix(Y_test, Y_pred_tree)
    print(confusion_matrix_tree)
    print(classification_report(Y_test, Y_pred_tree))

    #common standard to compare across models
    f1_clf = f1_score(Y_test, Y_pred_tree, average='binary')

    ##save model
    f_tree = open('tree_clf.pkl',"wb+")
    pickle.dump(tree_clf, f_tree)
    f_tree.close()

    f_tree_sc = open('tree_scaler.pkl',"wb+")
    pickle.dump(scaler, f_tree_sc)
    f_tree_sc.close()
    '''
    [[1801   27]
    [  62   61]]
           precision    recall  f1-score   support

        0       0.97      0.99      0.98      1828
        1       0.69      0.50      0.58       123

    avg / total       0.95      0.95      0.95      1951
    '''


    return tree_clf,f1_clf
```

Code: neural network 1/3

```
##2C Neural Network
##2Ci. Grid search that simulate the performance of different neural network design
def grid_search(X_train,X_test, Y_train,Y_test,num_training_sample):

    best_f1 = 0
    best_hidden_layers_list = []
    best_hidden_layers_tuple = ()
    #various depth
    for depth in range(1,5):
        print('Depth = '+str(depth))
        for layer_size in range(1,8):
            neuron_cnt = 0
            hidden_layers_list = []
            i = 0
            while i<depth:
                hidden_layers_list.append(layer_size)
                neuron_cnt += layer_size
                i+=1
            #pruning - to avoid over-training
            if num_training_sample<neuron_cnt:
                break

            hidden_layers_tuple = tuple(hidden_layers_list)
            nn_clf = MLPClassifier(alpha=1e-5,
                    hidden_layer_sizes=hidden_layers_tuple, random_state=1)

            nn_clf.fit(X_train,Y_train)
            Y_pred = nn_clf.predict(X_test)
            temp_f1 = f1_score(Y_test, Y_pred, average='binary')
            if temp_f1 > best_f1:
                best_f1 = temp_f1
                best_hidden_layers_list = hidden_layers_list
                best_hidden_layers_tuple = hidden_layers_tuple
    print(best_hidden_layers_list)
    return best_hidden_layers_list,best_hidden_layers_tuple
```

Code: neural network 2/3

```
#various size
# referencing: https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/
##2Cii. train network network
def train_NN(X,Y):
    print('Neural Network')
    #split the dataset into training set and testing set
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=.33, random_state=0)

    #preprocessing the data
    scaler = StandardScaler()
    scaler.fit(X_train)
    X_train = scaler.transform(X_train)
    X_test = scaler.transform(X_test)

    num_training_sample = len(X_train)
    best_hidden_layers_list,best_hidden_layers_tuple = grid_search(X_train, X_test, Y_train, Y_test,num_training_sample)
    nn_clf = MLPClassifier(alpha=1e-5,
                hidden_layer_sizes=best_hidden_layers_tuple, random_state=1)

    #fit the training data to the model
    nn_clf.fit(X_train,Y_train)

    ##metric 1: roc
    Y_score_nn = nn_clf.predict(X_test)
    fpr, tpr, thresholds = roc_curve(Y_test,Y_score_nn, pos_label=1)
    roc_auc = auc(fpr,tpr)
    lw=2
    plt.figure()
    plt.plot(fpr,tpr,color='darkorange',lw=lw,label='ROC curve (area = %0.2f)' %roc_auc)
    plt.plot([0,1],[0,1],color='navy',lw=lw,linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic - Neural Network')
    plt.legend(loc="lower right")
    plt.show()
```

Code: neural network 3/3

```
#various size
# referencing: https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/
##2Cii. train network network
def train_NN(X,Y):
    ....

    ##metric 2: Confusion matrix
    Y_pred_tree = nn_clf.predict(X_test)
    confusion_matrix_tree = confusion_matrix(Y_test, Y_pred_tree)
    print(confusion_matrix_tree)
    print(classification_report(Y_test, Y_pred_tree))

    #common standard to compare across models
    f1_clf = f1_score(Y_test, Y_score_nn, average='binary')

    ##save model
    f_nn = open('nn_clf.pkl',"wb+")
    pickle.dump(nn_clf, f_nn)
    f_nn.close()

    f_nn_sc = open('nn_scaler.pkl',"wb+")
    pickle.dump(scaler, f_nn_sc)
    f_nn_sc.close()

    '''
    [[1808   20]
     [  85   38]]
                 precision    recall  f1-score   support

              0       0.96      0.99      0.97      1828
              1       0.66      0.31      0.42       123

    avg / total       0.94      0.95      0.94      1951
    '''

    return nn_clf, f1_clf
```

Code: run all the functions above

```
'''
3. Run the functions above
'''
f1_list = []
f1_score_temp= 0
#logistic regression model
log_reg,f1_score_temp = train_logreg(X,Y)
f1_list.append(f1_score_temp)
log_reg.get_params()

#decision tree
tree_clf,f1_score_temp = train_tree(X,Y)
f1_list.append(f1_score_temp)
tree_clf.get_params()
#neural network
nn_clf,f1_score_temp = train_NN(X,Y)
f1_list.append(f1_score_temp)
nn_clf.get_params()
```

Code: visualize the results

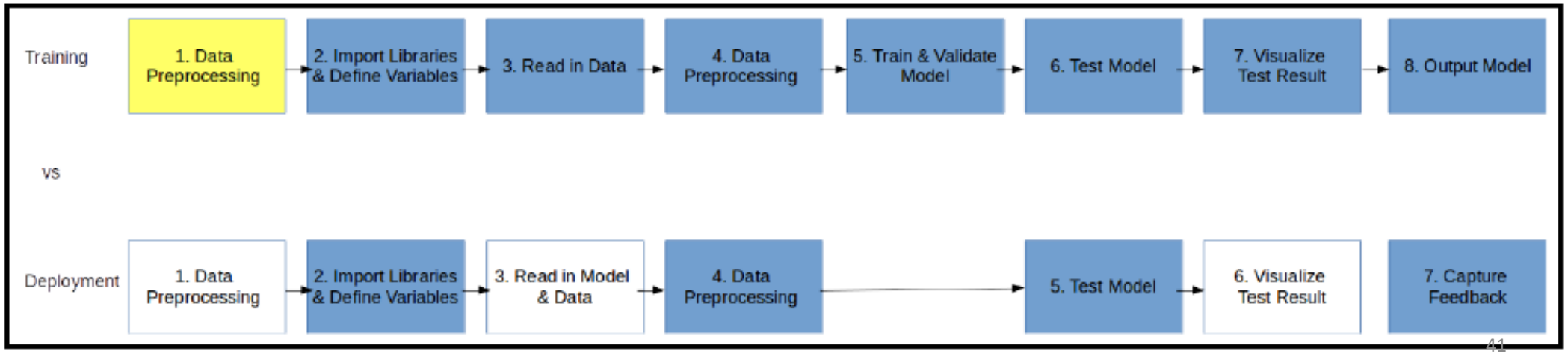
```
'''
#4 Visualize the result
'''
print('********************')
print('f1 of the models')
print(f1_list)
print('********************')

#for visualization of decision tree
x_feature_name = fields_list[:-1]
y_target_name = fields_list[-1]
d_tree_out_file = 'decision_tree'
dot_data = tree.export_graphviz(tree_clf, out_file=None,
                                feature_names=x_feature_name,
                                class_names=y_target_name,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render(d_tree_out_file)
```

Well done!

Congrats!

- You **have now delivered a model** that can be used at the operational level for predicting the chances of a borrower going bankrupt



Summary

- In this lecture, we learned about different AI modeling techniques to do classification with an example for predicting the chances of the borrower going bankrupt. The algorithms that we investigated were
  - logistic regression model,
  - decision trees,
  - and deep learning.

- We also learned about the various metrics of model performance :
  - ROC Curve
  - Confusion matrix
  - Classification report

- This is useful to validate if a model has learnt something and how to pick the best model