Statistics of Financial Markets

Studium Generale

Prof. Dr. Natalie Packham

Berlin School of Economics and Law

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Winter term 2024/25

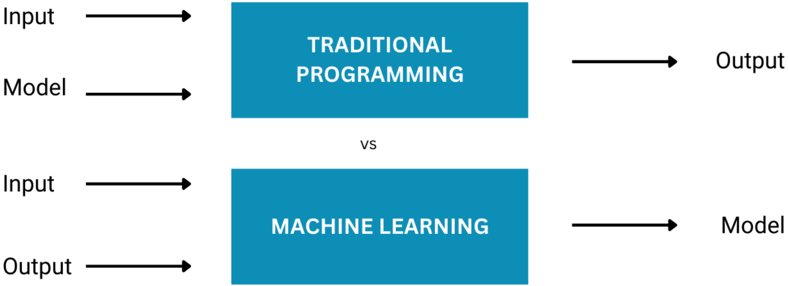
Contents

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance

Contents

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   5.1 Machine Learning Fundamentals
   5.2 Application areas of ML in Finance
   5.3 ML techniques
   5.4 Examples in Finance
   5.5 Explainable AI (XAI)

Introduction



Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
     - Model validation and measures of fit
     - Statistical methods in Machine Learning
     - Ridge regression, Lasso and Elastic Net
     - Logistic regression
     - Principal component analysis
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
     - Model validation and measures of fit
     - Statistical methods in Machine Learning
     - Ridge regression, Lasso and Elastic Net
     - Logistic regression
     - Principal component analysis
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Model validation and measures of fit

- In **Machine Learning** (Data Science, Statistical Learning, "Big Data", Artificial Intelligence, ...) there is no one single statistical method that performs *best* across all data sets.
- It is an important – and at times difficult – task to select the appropriate method or model for a given data set.
- We begin by studying a number of measures to assess the quality of fit, which in turn allows to compare methods and models.
- For more, see Chapters 2.2, 5.1 and 6.1.3 of (James *et al.*, 2013):
  - James, Witten, Hastie, Tibshirani: An Introduction to Statistical Learning. Springer, 2013.

Mean-square error and overfitting

- A commonly used measure for assessing how well predictions match observed data is the **mean squared error (MSE)**, which you know e.g. from ordinary least squares (OLS) in linear regression:

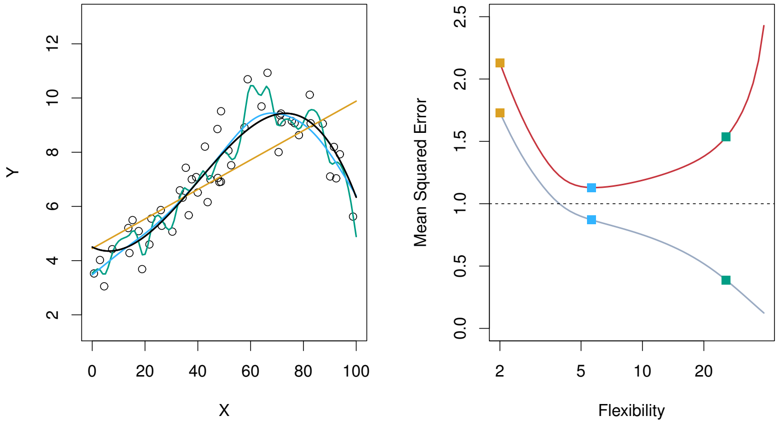
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

where $\hat{f}(x_i)$ is the prediction that the fitted method / model $\hat{f}$ gives for the i-th observation y_i.

Training and tests MSE

- In Linear Regression (a statistics method): whole data set is used for finding a linear function $\hat{f}$ that minimises the MSE.
- In ML: split the data set into a **training data set** and a **test data set**.
  - Reflects that we do not really care how well a method works on the training data.
  - Rather, we are interested in the accuracy of the prediction when applying the method to previously unseen data (the test data).
- In other words, first fit the training data to obtain the estimate $\hat{f}$, e.g. by minimising the MSE on the training data.
- Second, calculate the MSE on test data, which are data points that were not used in training.
- Choose method / model that gives lowest **test MSE** (or related measure).
- When **hyperparameters** (tuning parameters, e.g. number of hidden units in a neural network) are involved, a **validation data set** is added.

Test and training MSE



Source: James et al.: An Introduction to Statistical Learning. Springer, 2013.

Left: Test data shown as black dots, simulated from f (black smooth line).

Estimates of f: Linear regression (orange), smoothing splines (blue, green)

Right: Training MSE (grey), test MSE (red).

Flexibility denotes complexity of model (e.g. number of parameters)

The classification setting

- MSE applies in a regression setting.
- In a classification setting, we seek to estimate f on the basis of training observations $\{(x_1,y_1),(x_2,y_2),\dots,(x_n,y_n)\}$, where $y_1,\dots,y_n$ are qualitative.
- Here, the training **error rate**, which denotes the proportion of mistakes when applying the $\hat{f}$ to the training observations, is a measure of **accuracy**:

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{y_i\neq\hat{y}_i},$$

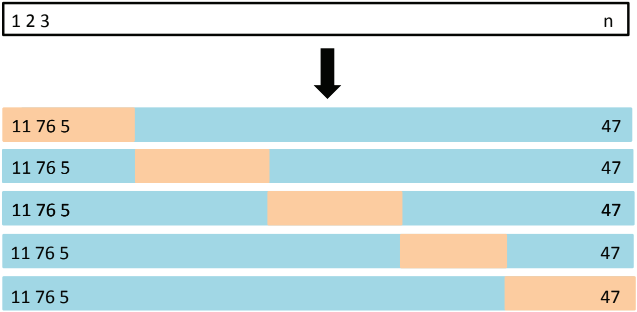
  where $\mathbf{1}_A$ is an *indicator function* taking value 1 if A is true and 0 otherwise.
- The **test error rate** is giving as the error rate from applying $\hat{f}$ to the test data set.

Cross validation

- **Cross validation (CV)** refers to several methods of building the test and training data sets.
- In k-fold CV, the data set is randomly divided in k groups or **folds** of approximately equal size.
- In k iterations, each first fold is treated as the test data set, while the $k-1$ other folds are taken as the training data.
- In this way, k MSE's of the test error are estimated and the k-fold CV estimate is given by

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i.$$

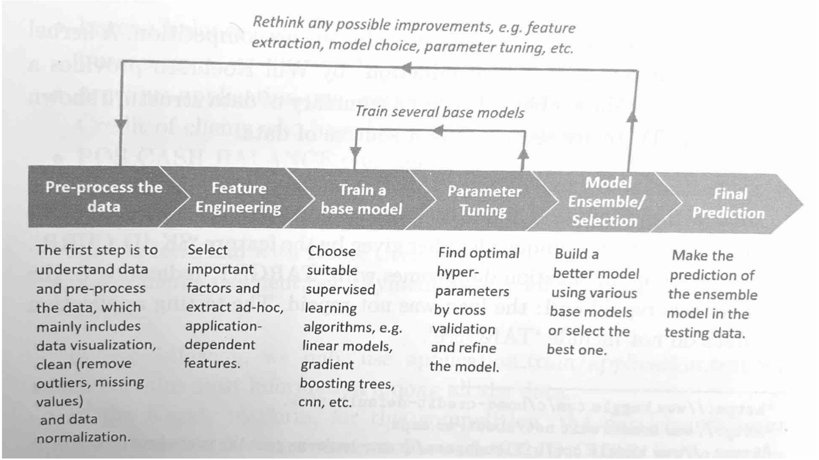
Cross validation



- A schematic display of 5-fold CV.
- A set of n observations is randomly split into five non-overlapping groups.
- Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue).
- The test error is estimated by averaging the five resulting MSE estimates.

ML pipeline

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law



Source: (Ni *et al.*, 2021)

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance

5.1 Machine Learning Fundamentals

Model validation and measures of fit

Statistical methods in Machine Learning

Ridge regression, Lasso and Elastic Net

Logistic regression

Principal component analysis

5.2 Application areas of ML in Finance

5.3 ML techniques

5.4 Examples in Finance

5.5 Explainable AI (XAI)

Statistical methods in ML

- We study a number of statistical methods that are popular in ML:
- ML methods are often classified into:
  - Regression vs. classification
  - Supervised vs. unsupervised learning (vs. reinforcement learning)
- The following methods illustrate these different aspects:
  - Ridge regression and Lasso (regression, supervised)
  - Logistic regression (classification, supervised)
  - Principal components analysis (PCA) (regression, unsupervised)

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance

5.1 Machine Learning Fundamentals

Model validation and measures of fit

Statistical methods in Machine Learning

Ridge regression, Lasso and Elastic Net

Logistic regression

Principal component analysis

5.2 Application areas of ML in Finance

5.3 ML techniques

5.4 Examples in Finance

5.5 Explainable AI (XAI)

Ordinary least squares (OLS)

- In linear regression, we assume a linear relationship between the **target** Y and the **feature vector** X:

$$Y = a + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m + \varepsilon,$$

  where $a, b_1, \dots, b_m$ are constants and ε is the error term.

- The ordinary least squares (OLS) estimates of a, b minimise the errors

$$\sum_{i=1}^{n} \varepsilon^2 = \sum_{i=1}^{n} (Y_i - a - b_1 X_{i1} - b_2 X_{i2} - \cdots - b_m X_{im})^2.$$

Regularisation

- In machine learning, especially when the number of features is high and when features are highly correlated, overfitting can easily occur.
- One way of dealing with this is known as **regularisation**.
- The most popular regularisation methods are:
  - Ridge regression
  - Lasso
  - Elastic net

Ridge regression

- **Ridge regression** is also known as **Tikhonov regularisation** and L_2 **regularisation**.
- Building on OLS, a term is added to the objective function that places a **penalty** on the size of the coefficients $b_1, \ldots, b_m$, by minimising:

$$\sum_{i=1}^{n}(Y_i - a - b_1 X_{i1} - b_2 X_{i2} - \cdots - b_m X_{im})^2 + \lambda \sum_{j=1}^{m} b_j^2.$$

- The constant λ is called **tuning parameter** or **hyperparameter** and controls the strength of the penalty factor.
- The term $\lambda \sum_{j=1}^{m} b_j^2$ is called the **shrinkage penalty**, as it will shrink the estimates of $b_1, \ldots, b_m$ towards zero.
- Selection of an appropriate value of λ can be achieved, for example, by **cross-validation**.

Ridge regression

- OLS estimates do not depend on the magnitude of the independent variables: multiplying X_j by a constant c leads to a scaling of the OLS-coefficient by $1/c$.
- This is different in regularised versions of regression: the estimated coefficients can change substantially when re-scaling independent variables.
- Therefore, it is custom, to **standardise** the features:

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}},$$

so that all variables are on the same scale, i.e., they all have a standard deviation of one.

Lasso

- **Lasso (Least absolute shrinkage and selection operator)**, also known as L_1 **regularisation** adds a different penalty:

$$\sum_{i=1}^{n}(Y_i - a - b_1X_{i1} - b_2X_{i2} - \cdots - b_mX_{im})^2 + \lambda\sum_{j=1}^{m}|b_j|.$$

- This has the interesting effect that the less relevant features are completely eliminated.
- For this reason, Lasso is also often used as a feature selection or variable selection method.

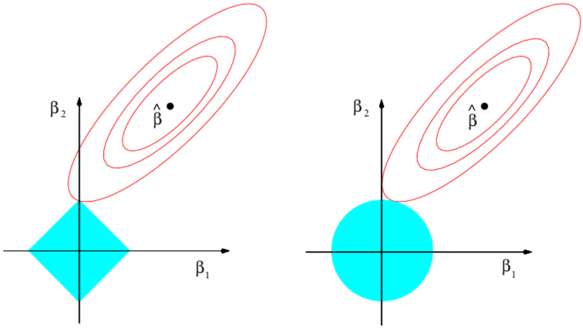
Elastic net regression

- **Elastic net regression** is a mixture of ridge regression and Lasso:

$$\sum_{i=1}^{n}(Y_i - a - b_1 X_{i1} - b_2 X_{i2} - \cdots - b_m X_{im})^2 + \lambda_1 \sum_{j=1}^{m} b_j^2 + \lambda_2 \sum_{j=1}^{m} |b_j|.$$

- Combining the effects of ridge regression and Lasso means that simultaneously
- some coefficients are reduced to zero (Lasso),
- some coefficients are reduced in size (ridge regression).

Illustration of regularisation constraints



Source: James et al.: An Introduction to Statistical Learning. Springer, 2013.

- $\hat{\beta}$: OLS solution
- Red level curves: residual sum of squares (RSS), error
- Turquoise areas: regularisation constraints (left: Lasso; right: ridge regression)

Example

- The following application predicts house prices based on different features of the property.
- The data set is from (Hull, 2021):
  Hull: Machine Learning in Business. 3rd edition, independently published, 2021.

House price example

```
>>> import pandas as pd # python's data handling package
>>> import numpy as np # python's scientific computing package
>>> import matplotlib.pyplot as plt # python's plotting package
>>> import seaborn as sns
>>> sns.set()

>>> from sklearn.metrics import mean_squared_error as mse
>>> from sklearn.model_selection import train_test_split
>>> # The sklearn library has cross-validation built in
>>> # https://scikit-learn.org/stable/modules/cross_validation.html
>>> from sklearn.model_selection import cross_val_score

>>> # Both features and target have already been scaled: mean = 0; SD = 1
>>> #data = pd.read_csv('data/Houseprice_data_scaled.csv')
>>> data = pd.read_csv('https://raw.githubusercontent.com/packham/' \
...                    + 'SFM/main/data/Houseprice_data_scaled.csv')

>>> type(data)
<class 'pandas.core.frame.DataFrame'>
```

House price example


```
>>> data.columns
Index(['LotArea', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'BsmtFinSF1', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF',
       'GrLivArea', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'TotRmsAbvGrd',
       'Fireplaces', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', 'Blmngtn', 'Blueste', 'BrDale', 'BrkSide', 'ClearCr',
       'CollgCr', 'Crawfor', 'Edwards', 'Gilbert', 'IDOTRR', 'MeadowV',
       'Mitchel', 'Names', 'NoRidge', 'NPkVill', 'NriddgHt', 'NWAmes',
       'OLDTown', 'SWISU', 'Sawyer', 'SawyerW', 'Somerst', 'StoneBr', 'Timber',
       'Veenker', 'Bsmt Qual', 'Sale Price'],
      dtype='object')
```

House price example


```
>>> X = data.drop('Sale Price', axis=1)
>>> y = data['Sale Price']
```

- sklearn can split training and testing data randomly.

```
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, \
...                                      test_size=0.25, random_state=123)
```

- If selecting a model and/or hyperparameters, use the training data set and possibly also a validation data set for training.
- Test data set is used to evaluate out-of-sample prediction error.
- Once model and hyperparameters are set, train on whole data set.

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression

>>> lr=LinearRegression()
>>> lr.fit(X,y)
LinearRegression()

>>> pred = lr.predict(X)
>>> mse(y, pred)
0.1202258814841823
```

```
>>> # Create dataFrame with corresponding feature and its respective coefficients

>>> coeffs = pd.DataFrame([['intercept'] + list(X.columns),[lr.intercept_] \
...           + lr.coef_.tolist()]).transpose().set_index(0)
```

```
>>> coeffs[0:24]
                                   1
0
intercept                -515.546024
LotArea                     0.086456
OverallQual                 0.212283
OverallCond                 0.090164
YearBuilt                   0.160025
YearRemodAdd                0.026118
BsmtFinSF1                  0.056078
BsmtUnfSF                  -0.082941
TotalBsmtSF                 0.174829
1stFlrSF                   -0.037159
2ndFlrSF                    -0.09013
GrLivArea                   0.440879
FullBath                   -0.034017
HalfBath                    0.031653
BedroomAbvGr               -0.036824
TotRmsAbvGrd                0.001981
Fireplaces                  0.039034
GarageCars                  0.032289
GarageArea                  0.086777
WoodDeckSF                  0.026919
OpenPorchSF                 0.034702
EnclosedPorch              -0.005115
Blmngtn        -599427595824.349487
Blueste        -276192024995.470886
```

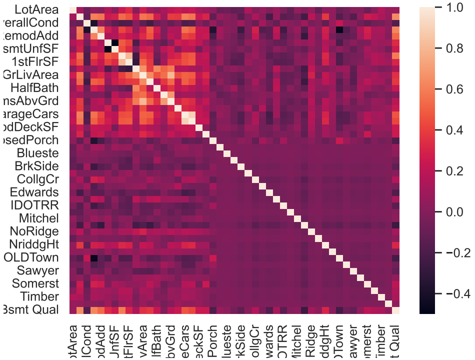
```
>>> coeffs[24:]
                                  1
0
BrDale       -614827056936.071533
BrkSide     -1110362165388.210205
ClearCr      -775075647820.064819
CollgCr     -1659751586060.211182
Crawfor     -1009501649342.026123
Edwards     -1404971994187.303955
Gilbert     -1398984894299.677246
IDOTRR        -864598150253.17041
MeadowV      -629833264199.754272
Mitchel     -1069667259856.681519
Names       -2183000890067.655518
NoRidge      -935336450765.461548
NPkVill       -567320013921.18689
NriddgHt     -1386910194091.9646
NWAmes      -1285049035739.846924
OLDTown     -1555372372895.441162
SWISU        -786870415424.133667
Sawyer      -1257958402207.644287
SawyerW      -1201522625572.22168
Somerst     -1468743662666.667725
StoneBr      -738475772927.898315
Timber       -875090200691.151245
Veenker      -533203156923.734131
Bsmt Qual                0.033724
```

Linear Regression

- Observe how the OLS coefficients are all non-zero.
- Some coefficients are negative where a positive coefficient would be expected.
- Some coefficients are unreasonably large.
- This is an indication that the model is struggling to fit the large number of features.

- Indeed, some correlations are high, as the heatmap indicates, which may cause ill-fitting.

```
>>> sns.heatmap(X_train.corr())
<Axes: >
>>> plt.savefig('_pics/heatmap.pdf')
```



- Likewise some correlations of the sale price with the features are close to zero:

```
>>> plt.clf()
>>> sns.heatmap(pd.DataFrame(data.corr().iloc[:,-1]))
<Axes: >
>>> plt.savefig('_pics/heatmap2.pdf')
```

Ridge regression

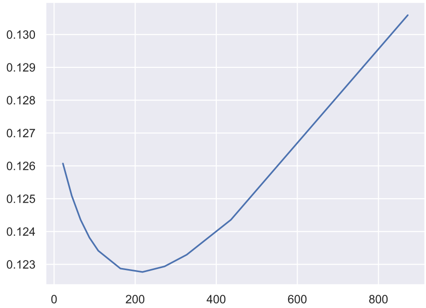
- Use cross-validation for training and testing.
- Specify the number of folds (‘cv‘) and MSE as the ‘scoring‘ function.
- On each fold, train model and determine test MSE:

```
>>> # Importing Ridge
>>> from sklearn.linear_model import Ridge

>>> n=int(len(X)*.75) # choose 75% of data for training (=4 folds)

>>> # The alpha used by Python's ridge should be the lambda
>>> # times the number of observations
>>> alphas=[0.01*n, 0.02*n, 0.03*n, 0.04*n, 0.05*n, 0.075*n,0.1*n,0.125*n, \
...         0.15*n,0.2*n, 0.4*n]
>>> mses=[]
>>> for alpha in alphas:
...     scores = cross_val_score(Ridge(alpha=alpha), X, y, cv=4, \
...                              scoring="neg_root_mean_squared_error")**2
...     mses.append(np.mean(scores))
...
```

Ridge regression

- Average test MSE varies with the hyperparameter α.

```
>>> alpha0 = alphas[np.argmin(mses)]
>>> alpha0
218.10000000000002

>>> plt.clf()
>>> plt.plot(alphas, mses)
[<matplotlib.lines.Line2D object at 0x16e9c60d0>]
>>> plt.savefig("_pics/mses_ridge.pdf")
```



Ridge regression

- Once optimal parameter α has been found, train on whole data set.

```
>>> ridge=Ridge(alpha=alpha0)
>>> ridge.fit(X,y)
Ridge(alpha=218.10000000000002)
>>> pred=ridge.predict(X)
>>> mse(y, pred)
0.11462514988978675

>>> coeffs = pd.DataFrame([['intercept'] + list(X.columns),[ridge.intercept_] \
...                          + ridge.coef_.tolist()]).transpose().set_index(0)
```

Ridge regression


```
>>> coeffs[0:24]
                       1
0
intercept       0.013747
LotArea         0.077606
OverallQual     0.206007
OverallCond     0.072271
YearBuilt       0.111912
YearRemodAdd    0.050475
BsmtFinSF1      0.098782
BsmtUnfSF      -0.030174
TotalBsmtSF     0.114397
1stFlrSF        0.131309
2ndFlrSF        0.089828
GrLivArea       0.174094
FullBath        0.006225
HalfBath        0.025541
BedroomAbvGr   -0.057019
TotRmsAbvGrd     0.05657
Fireplaces      0.039576
GarageCars      0.024692
GarageArea       0.07656
WoodDeckSF      0.025857
OpenPorchSF     0.020792
EnclosedPorch   0.003155
Blmngtn        -0.014768
Blueste         -0.01113
```

```
>>> coeffs[24:]
                  1
0
BrDale    -0.021509
BrkSide    0.010937
ClearCr   -0.007176
CollgCr   -0.009297
Crawfor    0.033355
Edwards   -0.007398
Gilbert   -0.014602
IDOTRR    -0.002863
MeadowV   -0.016157
Mitchel   -0.029205
Names     -0.031012
NoRidge    0.059522
NPkVill   -0.022473
NriddgHt   0.114833
NWAmes    -0.048128
OLDTown   -0.023063
SWISU     -0.009618
Sawyer    -0.016464
SawyerW   -0.026844
Somerst    0.024684
StoneBr    0.078721
Timber     0.007753
Veenker   -0.004438
Bsmt Qual  0.041138
```

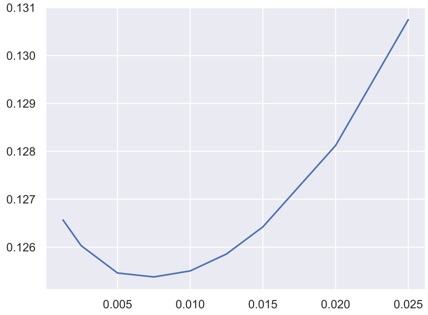
Lasso

- First, find the parameter with minimal average test MSE.

```
>>> from sklearn.linear_model import Lasso

>>> # The alphas are half the lambdas
>>> alphas=[0.0025/2, 0.005/2, 0.01/2, 0.015/2, 0.02/2, 0.025/2, \
...         0.03/2,0.04/2, 0.05/2]
>>> mses=[]
>>> for alpha in alphas:
...     scores = cross_val_score(Lasso(alpha=alpha), X, y, \
...                              cv=4, scoring="neg_root_mean_squared_error")**2
...     mses.append(np.mean(scores))
...
>>> alpha0=alphas[np.argmin(mses)]
>>> alpha0
0.0075
```

Lasso


```
>>> plt.clf()
>>> plt.plot(alphas, mses)
[<matplotlib.lines.Line2D object at 0x16ea1dbd0>]
>>> plt.savefig("_pics/mses_lasso.pdf")
```



Lasso

- Now train on optimal α:

```
>>> lasso=Lasso(alpha=alpha0)
>>> lasso.fit(X,y)
Lasso(alpha=0.0075)
>>> pred=lasso.predict(X)
>>> mse(y, pred)
0.11561861295807133

>>> coeffs = pd.DataFrame([['intercept'] + list(X.columns),[lasso.intercept_] \
...                        + lasso.coef_.tolist()]).transpose().set_index(0)
```

- Some of the coefficients have now been set to zero.
- Lasso acts as a variable selection method.

Lasso


```
>>> coeffs[0:24]
                      1
0
intercept      0.014318
LotArea        0.081478
OverallQual    0.239335
OverallCond    0.067869
YearBuilt       0.12548
YearRemodAdd   0.044188
BsmtFinSF1     0.127284
BsmtUnfSF          -0.0
TotalBsmtSF    0.090746
1stFlrSF       0.041477
2ndFlrSF            0.0
GrLivArea       0.33046
FullBath           -0.0
HalfBath       0.008588
BedroomAbvGr  -0.051891
TotRmsAbvGrd   0.020714
Fireplaces      0.02929
GarageCars          0.0
GarageArea     0.088357
WoodDeckSF     0.019206
OpenPorchSF     0.01303
EnclosedPorch       0.0
Blmngtn       -0.001796
Blueste       -0.004187
```

```
>>> coeffs[24:]
                 1
0
BrDale   -0.009219
BrkSide   0.016484
ClearCr        0.0
CollgCr        0.0
Crawfor   0.039088
Edwards   0.000074
Gilbert        0.0
IDOTRR    0.000669
MeadowV  -0.002043
Mitchel  -0.011964
Names    -0.002662
NoRidge   0.059195
NPkVill  -0.008885
NriddgHt  0.123114
NWAmes   -0.030252
OLDTown  -0.001102
SWISU         -0.0
Sawyer        -0.0
SawyerW  -0.010306
Somerst   0.026829
StoneBr   0.078655
Timber    0.008663
Veenker        0.0
Bsmt Qual 0.026486
```

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
     - Model validation and measures of fit
     - Statistical methods in Machine Learning
     - Ridge regression, Lasso and Elastic Net
     - Logistic regression
     - Principal component analysis
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Logistic regression

- In a regression setting, numerical variables are predicted.
- Another application is **classification**, which is about predicting the category a new observation belongs to.
- In supervised learning, and with two categories, a variation of regression, called **logistic regression** can be used.
- Given features $X_1, \dots, X_m$, suppose there are two classes to which observations can belong.
- An example is the prediction of a loan's default risk, given characteristics of the creditor such as age, education, marital status, etc.
- Another example is the classification of e-mails into junk or non-junk e-mails.

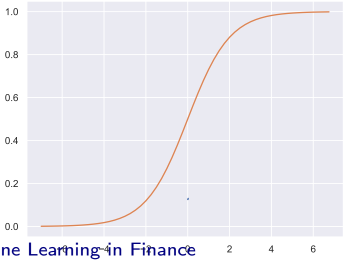
Logistic regression

- Logistic regression can be used to calculate the probability of a positive outcome via the **sigmoid function**

$$P(Y=1|X) = \frac{1}{1+e^{-X}} = \frac{e^X}{1+e^X},$$

where e is the Euler constant.

```
>>> x=np.arange(-7,7,0.25)
>>> plt.plot(x, 1/(1+np.exp(-x)))
[<matplotlib.lines.Line2D object at 0x29a571810>]
>>> plt.savefig("_pics/sigmoid.pdf")
```



Logistic regression

- Setting $X = a + b_1 X_1 + b_2 X_2 + \cdots + b_m X_m$, the probability of a positive outcome is

$$P(Y = 1 | X_1, \ldots, X_m) = \frac{1}{1 + \exp(-a - \sum_{j=1}^{m} b_j X_j)}.$$

- The objective is to find the coefficients $a, b_1, \ldots, b_m$ that best classify the given data.
- **Maximum likelihood** is a versatile method for this type of problem, when OLS does not apply.
- Without going into detail, the **log likelihood function** is given as

$$\ell(a, b_1, \ldots, b_m | x_1, \ldots, x_n) = \sum_{k:y_k=1} \ln(p(x_k)) + \sum_{k:y_k=0} \ln(1 - p(x_k)),$$

and the parameters are chosen that maximise this function.

- (Note: The likelihood function is derived by considering the observations to be independent outcomes of a Bernoulli random variable.)

Example: Credit risk

- The dataset in this example is taken from (James *et al.*, 2013):
  James et al.: An Introduction to Statistical Learning. Springer, 2013.
- It contains simulated data of defaults on credit card payments, on the basis of credit card balance (amongst other things).
- An excellent tutorial with examples on logistic regression in Python is available here: `https://realpython.com/logistic-regression-python/`.
- We will use the `sklearn` package below. Logistic regression can also be performed with the `statsmodels.api`, in which case p-values and other statistics are calculated.

Example: credit risk

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> import pandas as pd
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.metrics import classification_report, confusion_matrix
>>> from sklearn.model_selection import train_test_split

>>> data = pd.read_csv("https://raw.githubusercontent.com/packham/" \
...                    + "SFM/main/data/Default_JamesEtAl.csv")

>>> data.head()
  default student      balance        income
0      No      No   729.526495  44361.625074
1      No     Yes   817.180407  12106.134700
2      No      No  1073.549164  31767.138947
3      No      No   529.250605  35704.493935
4      No      No   785.655883  38463.495879
```

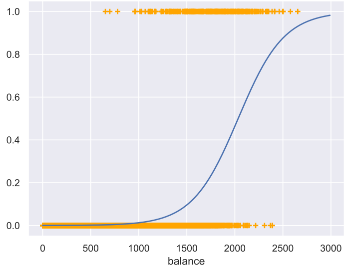
Example: credit risk


```
>>> x=np.array(data["balance"]).reshape(-1,1) # array must be two-dimensional
>>> y=np.array([True if x=="Yes" else False for x in data["default"]])
>>> x_train, x_test, y_train, y_test \
...     = train_test_split(x, y, test_size=0.2, random_state=0)

>>> model = LogisticRegression(solver='liblinear', random_state=0)
>>> model.fit(x_train,y_train)
LogisticRegression(random_state=0, solver='liblinear')


>>> # fitted parameters
>>> a=model.intercept_[0]
>>> b=model.coef_[0,0]
>>> [a,b]
[-8.537515117344613, 0.004196083866542488б]
```

Example: credit risk

- Scatter plot of data and fitted logistic function:

```
>>> plt.clf()
>>> plt.scatter(x,y,c='orange', marker="+")
<matplotlib.collections.PathCollection object at 0x29b2f8f90>
>>> plt.xlabel('balance')
Text(0.5, 0, 'balance')
>>> xrange=range(0,3000,10)
>>> plt.plot(xrange,1/(1+np.exp(-a-b *xrange)))
[<matplotlib.lines.Line2D object at 0x29b361010>]
>>> plt.savefig("_pics/logistic.pdf")
```



Example: credit risk

- Predictions:

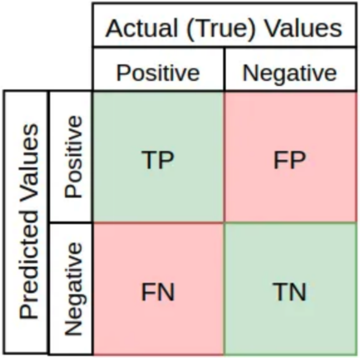
```
>>> model.predict_proba(x_train)[:5]
array([[0.98633215, 0.01366785],
       [0.98600606, 0.01399394],
       [0.98133591, 0.01866409],
       [0.998236  , 0.001764  ],
       [0.99643573, 0.00356427]])
```

- Mean accuracy of the model:

```
>>> [model.score(x_train,y_train), model.score(x_test,y_test)]
[0.972875, 0.968]
```

Example: credit risk

- Confusion matrix:



https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec

```
>>> confusion_matrix(y_test,model.predict(x_test))
array([[1923,    3],
       [  61,   13]])
```

Example: credit risk

- Classification report:

```
>>> print(classification_report(y_test, model.predict(x_test)))
              precision    recall  f1-score   support

       False       0.97      1.00      0.98      1926
        True       0.81      0.18      0.29        74

    accuracy                           0.97      2000
   macro avg       0.89      0.59      0.64      2000
weighted avg       0.96      0.97      0.96      2000
```

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

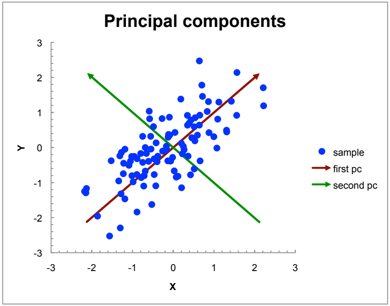
5. Introduction to Machine Learning in Finance
   5.1 Machine Learning Fundamentals
      Model validation and measures of fit
      Statistical methods in Machine Learning
      Ridge regression, Lasso and Elastic Net
      Logistic regression
      Principal component analysis
   5.2 Application areas of ML in Finance
   5.3 ML techniques
   5.4 Examples in Finance
   5.5 Explainable AI (XAI)

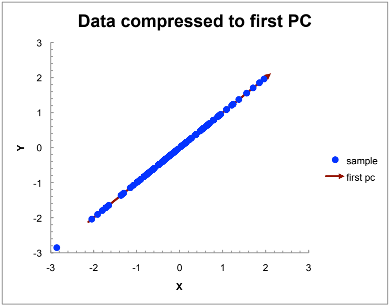
Principal Component Analysis

- **Principal Component Analysis (PCA)** summarises a large set of correlated variables by smaller number of representative variables that explain most of the variability of the original data set.
- **PCA** is a standard method for reducing the dimension of high dimensional, highly correlated systems.
- **PCA** is a particular rotation of the axes, driven by random variables or data.
- Key idea is to align random variables / data such that
  - first dimension captures maximal variance,
  - second dimension is orthogonal and captures second-most variance,
  - etc.

Principal Component Analysis

- Example:





Principal Component Analysis

- See James *et al.* (2013), Section 10.2, for the following.
- Given $n \times d$ data set $\mathbf{X}$ that is **standardised**.
- **First principal component**: find **scores**

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{d1}x_{id}, \quad i = 1, \ldots, n,$$

  that have largest sample variance, subject to constraint $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- In other words, **first PC vector** solves optimisation problem

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \underbrace{\left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2}_{=z_{i1}^2} \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

- Second (and higher) PCs: linear combination of data uncorrelated with first PC(s) and with largest variance (subject to constraint).

Principal Component Analysis

- **Principal components (PCs)** are the **eigenvectors** of covariance / correlation matrix.
- **Eigenvalues** express **amount of variance** captured by each PC.
- Compact notation (recall that $\mathbf{X}$ is standardised):

$$\mathbf{Z} = \mathbf{\Phi}^T \mathbf{X}$$

- PCs can be viewed as factors, giving factor model

$$\mathbf{X} = \mathbf{\Phi}\,\mathbf{Z}.$$

- $\mathbf{\Phi}$ are the eigenvectors of correlation matrix of $\mathbf{X}$.

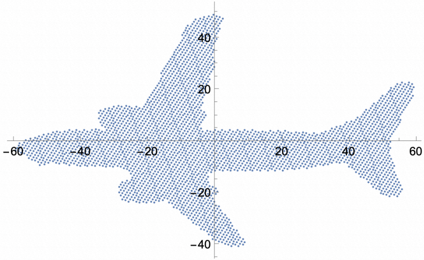
- Example from Mathematica:

In[264]:=

`shape = Position[ImageData@`  `, 1, {2}];`

```
ListPlot[PrincipalComponents[N@shape]]
```

Out[265]=



Contents

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   5.1 Machine Learning Fundamentals
   5.2 Application areas of ML in Finance
   5.3 ML techniques
   5.4 Examples in Finance
   5.5 Explainable AI (XAI)

Application areas of ML in Finance

- **Risk Management**: Assess and manage financial risks by analyzing market trends, economic indicators, and historical data to predict potential risks and advise on mitigation strategies.

  E.g. XAIFi (Explainable AI for the Risk Management of FinTech) project, https://www.ifaf-berlin.de/projekte/xaifi/; Stress-testing (Packham, 2024; Packham and Woebbeking, 2023)

- **Credit Scoring and Loan Underwriting**: Assess credit quality of applicants by analyzing a wide range of data, including credit history, transaction behaviors, and even social media activity.

  See e.g. (Dumitrescu *et al.*, 2022) and references herein

- **Fraud Detection and Prevention**: Identify unusual patterns and anomalies that may indicate fraudulent activities. These systems analyze transaction data in real-time to detect and prevent fraud ($\mapsto$ Buy-now-pay-later (BNPL)).

  https://en.wikipedia.org/wiki/Data_analysis_for_fraud_detection

- **Anti-Money Laundering (AML)**: Monitor transactions and identify patterns indicative of money laundering, ensuring compliance with regulatory standards.

  (Han *et al.*, 2020; Jensen and Iosifidis, 2023)

Application areas of ML in Finance

- **Calibration, Pricing and Hedging**: Model calibration, rapid derivatives pricing and hedging on the trading floor.

  (Funahashi, 2024; Ruf and Wang, 2020; Büchel *et al.*, 2021; Buehler *et al.*, 2019; De Spiegeleer *et al.*, 2018)

- **Stock Market Prediction**: Analyse historical stock data, market sentiment, and other relevant factors to predict future stock prices, aiding investors in making informed decisions.

  https://en.wikipedia.org/wiki/Stock_market_prediction#Machine_learning; also fractional trading in XAIFi project

- **Algorithmic Trading**: Trading algorithms that analyze market data, predict price movements, and execute trades at optimal times, often within milliseconds.

  https://en.wikipedia.org/wiki/Stock_market_prediction#Machine_learning

Application areas of ML in Finance

- **Portfolio Management (Robo-Advisors)**: Automated investment platforms, known as robo-advisors, use machine learning to create and manage investment portfolios tailored to individual preferences and risk tolerances.
  Literature review: (Torno *et al.*, 2021)
- **Customer Service Chatbots**: AI-driven chatbots handle customer inquiries, provide account information, and assist with transactions.
  https://www.consumerfinance.gov/data-research/research-reports/chatbots-in-consumer-finance/chatbots-in-consumer-finance/

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
     - Decision trees
     - Random forests
     - Gradient Boosting Decison Trees
     - Neural networks
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

ML techniques

- This section primarily uses material from:
- (Ni *et al.*, 2021)

  Ni, H., X. Dong, J. Zheng, and G. Yu. An Introduction to Machine Learning in Quantitative Finance. World Scientific, 2021.
- (Bernard, 2021)

  Bernard, E. Introduction to Machine Learning. Wolfram Media, 2021.

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

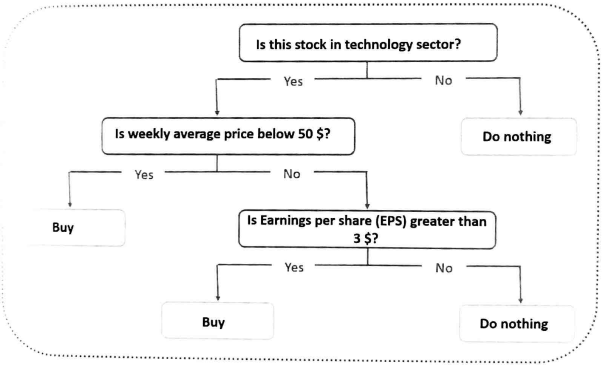
3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
     - Decision trees
     - Random forests
     - Gradient Boosting Decison Trees
     - Neural networks
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Decision trees

- A **decision tree** is a undirected graph, such that for any two nodes there exists only one path connecting them.
- Fixing root of tree makes it a directed graph (like a flowchart)
- Terminology: root node, decision node, leaf/terminal node, parent and child node



Source: (Ni *et al.*, 2021)

Decision trees

- Pros:
  - Easy to understand (interpretability)
  - Useful in data exploration and for feature selection
  - Data type is not a constraint
  - Non-parametric method
- Cons:
  - Over-fitting
  - Not suitable for continuous variables
- Most decision tress are **binary trees**:
  - Each decision node has a splitting rule that assigns observations to either left or right child nodes.
  - Terminal node thus identifies a partition of the observation space according to splitting rules.

Regression trees

Regression tree:

- Tree with leaves $R_1, \dots, R_m$ gives rise to

$$f(x) = \sum_{m=1}^{M} c_m \, \mathbf{1}_{\{x \in R_m\}},$$

  where $c_1, \dots, c_M$ are constants.

- Given $R_1, \dots, R_m$, finding optimal regression tree boils down to minimising:

$$\sum_{i=1}^{n} (y_i - f(x_i))^2.$$

- Main problem is to find global optimal partition.

Classification trees

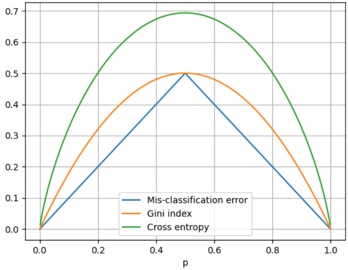
Classification tree:

- Percentage of samples estimated as class k in region R_m:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \mathbf{1}_{\{y_i \in k\}},$$

  with N_m number of samples belonging to region R_m.

- **Impurity measures**: performance measures:
  - **Missclassification error**: $\sum_{m=1}^{M}(1 - \hat{p}_{mm}$
  - **Gini index**: $\sum_{k=1}^{M} \hat{p}_{mk}(1 - \hat{p}_{mk})$
  - **Cross entropy**: $-\sum_{k=1}^{M} \hat{p}_{mk} \log(\hat{p}_{mk})$
- **Feature importance**:
  - e.g. impurity measure gain



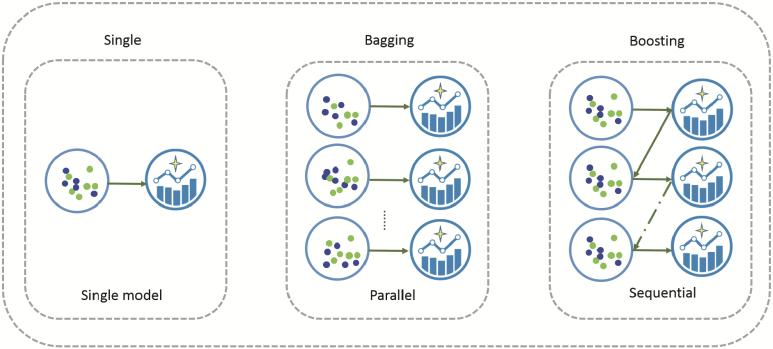
Source: (Ni *et al.*, 2021)

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
     - Decision trees
     - Random forests
     - Gradient Boosting Decison Trees
     - Neural networks
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Ensemble methods

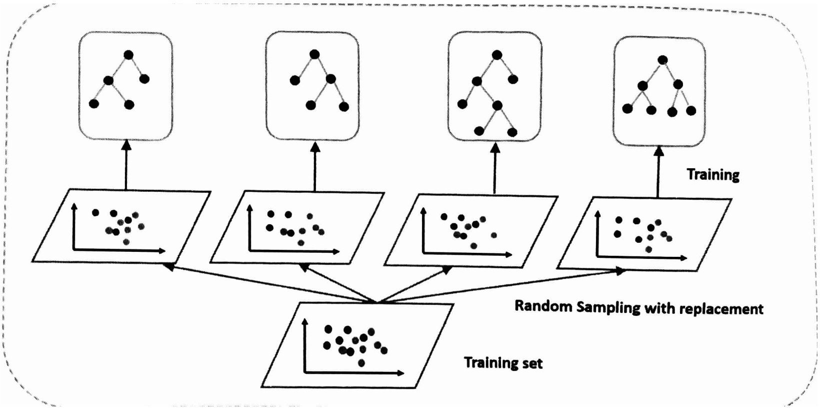


Source: (Ni *et al.*, 2021)

Random Forest

- A **forest** is an undirected graph, all of whose connected components are trees. In other words, the graph consists fo a disjoint union of trees.
- **Random forests** (Breiman, 2001) generate multiple decision trees by
  - random sampling of data ("bootstrap aggregation");
  - random selection of input features for generating individual base decision trees.
- Prediction is average model prediction (regression) or majority vote (classification).
- Variance of error of random forest models reduced by
  - predictive power of individual decision trees, and
  - correlation among trees.

Random forest



Source: (Ni *et al.*, 2021)

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
     - Decision trees
     - Random forests
     - Gradient Boosting Decison Trees
     - Neural networks
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

Gradient Boosting Decison Trees

- Main idea of **boosting**: Update subsequent predictors based on the error of previous predictors.
- **Gradient boosting**:
  - Goal is to minimise loss function $L(f, y)$.
  - Sequence of weak learners (here: decision tree) h_m, $m = 1, \ldots, M$, is used to build sequence of predictors f_m, $m = 1, \ldots, M$.
  - Updates uses gradient, similar to **gradient descent** (a popular optimisation algorithm) for optimal "direction".

Gradient boosting

Algorithm 2: Gradient Boosting Algorithm

Input: $(x_i, y_i)_{i=1}^{N}$.
Initialize f_0 by a constant γ_0 via the following equation:

$$\gamma_0 = \arg\min_{\gamma} L(y, \gamma);$$

for $m = 1 : M$ **do**
  for $i = 1 : N$ **do**
    Compute the residuals

$$r_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

  end for
  Fit a base model learner h_m to the target r_{im}, using the data $(x_i, r_{im})_{i=1}^{n}$.
  Solve the one dimensional optimization problem

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i)).$$

  Update f_m using the following formula:

$$f_m(x) = f_{m-1}(x) + \gamma_m h_m(x).$$

end for
Output: f_M.

Source: (Ni *et al.*, 2021)

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

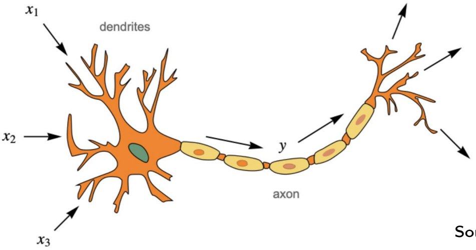
4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
     - Decision trees
     - Random forests
     - Gradient Boosting Decison Trees
     - Neural networks
   - 5.4 Examples in Finance
   - 5.5 Explainable AI (XAI)

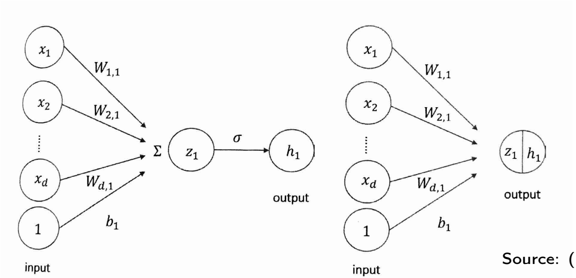
Neural networks – basic concepts

- A **neuron** is the basic unit of a neural network.
- Neurons are connected and transmit **signals** between each other.
- The input (z_1 on the following slide) represents the input received by an (artificial) neuron.
- This is transformed into the output h_1 by an **activation function** σ.
- Several parallel neurons constitute a **layer**.
- Several (connected) layers constitute a multi-layer neural network.
- **Learning** refers to finding the optimal parameters of the network.
- Types of layers:
  - input layers
  - hidden layers
  - output layers

Basic concepts: Neural networks



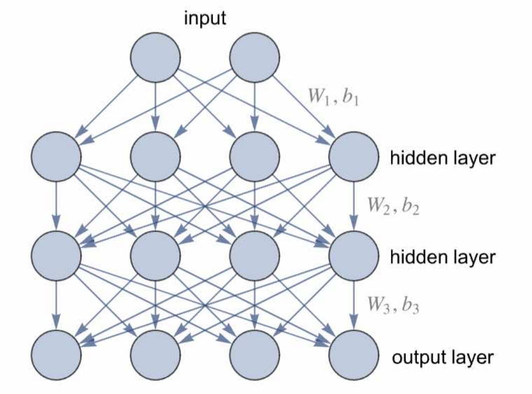
Source: (Bernard, 2021)



Source: (Ni *et al.*, 2021)

Basic concepts: Deep neural network

- A **deep neural network** has several hidden layers

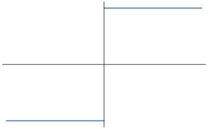


Source: (Bernard, 2021)

Basic concepts: Activation functions

- The **perceptron** is a neuron that acts as a binary linear classifier:

$$f(x) = \operatorname{sign}(Wx + b).$$

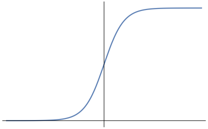


- **Heaviside function (step function)**:

$$f(x) = \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x = -1. \end{cases}$$

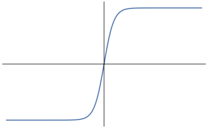
- **Sigmoid function**:

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$



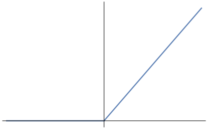
- **Tanh** as a generalisation of σ:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$



- **ReLU** (rectified linear unit):

$$f(x) = \max(x, 0).$$



- **Softmax** (normalised exponential function):

$$f(x_1, \dots, x_n) = \frac{1}{\sum_{i=1}^{n} e^{x_i}} \left(e^{x_1}, \dots, e^{x_n}\right)$$

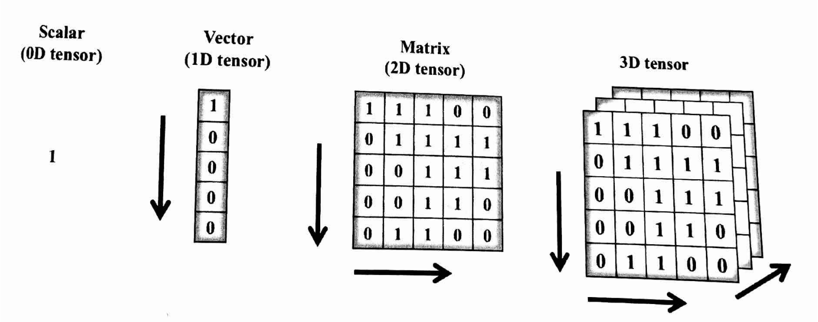
This is often used to model probability distributions and popular for classification problems.

- **Identity function**:

$$f(x) = x.$$

Basic concepts: Tensors

- All computations in neural networks boil down to **tensors**:

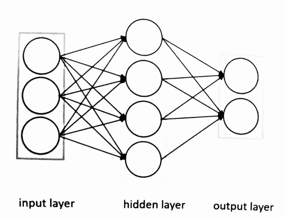


Source: (Ni *et al.*, 2021)

Shallow neural networks

- **Shallow neural network** (2-layer artificial NN) has one hidden layer:
  - Input layer ($h^{(0)} : \mathbb{R}^d \to \mathbb{R}^d$) (identity):

$$x = \left(x^{(1)}, x^{(2)}, \dots, x^{(d)}\right) \mapsto x$$



  - Hidden layer ($h^{(1)} : \mathbb{R}^d \to \mathbb{R}^{n_1}$):

$$z^{(1)} = W^{(1)}x + b^{(1)}$$
$$h^{(1)} = \sigma_1(z^{(1)}),$$

where $W^{(1)}$ is $n_1 \times d$, $b^{(1)}$ is n_1-dimensional;

  - Output layer ($h^{(2)} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$):

$$z^{(2)} = W^{(2)}h^{(1)} + b^{(2)}$$
$$h^{(2)} = \sigma_2(z^{(2)}),$$

E.g. identity for regression, Softmax for classification

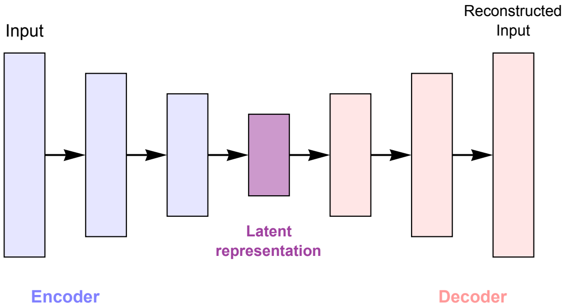
where $W^{(2)}$ is $n_2 \times n_1$, $b^{(2)}$ is n_2-dimensional.

Shallow neural networks

- For non-linear regression, sigmoid function is natural choice for σ_1.
- One can show that a shallow NN is a **universal function approximator**, meaning it can model any suitably smooth function, given enough hidden units, to any desired level of accuracy, see e.g. (Hornik *et al.*, 1989; Cybenko, 1989; Hornik, 1991).

A simple NN: Autoencoder

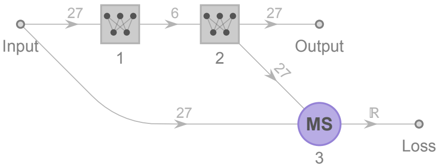
- An **autoencoder (AE)** is a simple feed forward neural network used for dimension reduction.
- Generalising PCA, it can capture nonlinear relationships.
- See e.g. (Bourlard and Kamp, 1988; Kramer, 1991; Hinton and Salakhutdinov, 2006).
- Schematic autoencoder representation:[22]



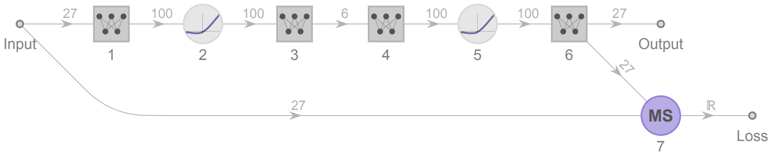
[22] https://reference.wolfram.com/language/ref/method/Autoencoder.html

A simple NN: Autoencoder

- PCA as AE:



- Optimal AE from training various architectures on test/validation data sets.



Source: (Packham, 2024)

Contents

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
    - 5.1 Machine Learning Fundamentals
    - 5.2 Application areas of ML in Finance
    - 5.3 ML techniques
    - 5.4 Examples in Finance
        - Credit risk modelling
        - Algorithmic trading
    - 5.5 Explainable AI (XAI)

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
   - 5.4 Examples in Finance
     - Credit risk modelling
     - Algorithmic trading
   - 5.5 Explainable AI (XAI)

Kaggle competition: Home Credit Default Risk

- A good starting point for examples in credit risk is the **Kaggle competition** on **Home Credit Default Risk**.
- **Home Credit** launched two competitions (2018 and 2024).
- 2018 competition:
  - More than 8000 participants and a total of $70,000 in prizes.
  - https://www.kaggle.com/competitions/home-credit-default-risk/
  - https://www.kaggle.com/code/willkoehrsen/start-here-a-gentle-introduction
- 2024 competition:
  - More than 5000 participants and a total of $105,000 in prizes.
  - https://www.kaggle.com/competitions/home-credit-credit-risk-model-stability/
  - https://www.kaggle.com/code/jetakow/home-credit-2024-starter-notebook

Contents

1. Review of Statistical Foundations

2. Regression and time series analysis

3. Derivatives pricing and hedging

4. Option pricing with Monte Carlo simulation

5. Introduction to Machine Learning in Finance
   - 5.1 Machine Learning Fundamentals
   - 5.2 Application areas of ML in Finance
   - 5.3 ML techniques
   - 5.4 Examples in Finance
     - Credit risk modelling
     - Algorithmic trading
   - 5.5 Explainable AI (XAI)

- Open Quant League Trading Competition:
- https://www.quantconnect.com/league/
- Open source, but not necessarily ML algorithms.

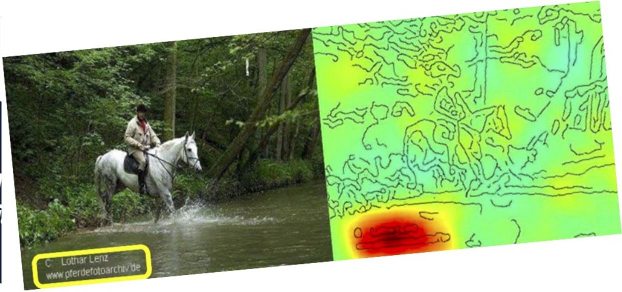
Contents

1. Review of Statistical Foundations
2. Regression and time series analysis
3. Derivatives pricing and hedging
4. Option pricing with Monte Carlo simulation
5. Introduction to Machine Learning in Finance
   5.1 Machine Learning Fundamentals
   5.2 Application areas of ML in Finance
   5.3 ML techniques
   5.4 Examples in Finance
   5.5 Explainable AI (XAI)

Why explainability?

https://www.technologyreview.com/2018/10/10/139858/amazon-ditched-ai-recruitment-software-because-it-was-biased-against-women/

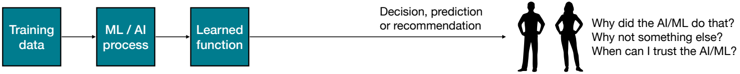
Why explainability?



(Lapuschkin *et al.*, 2019)

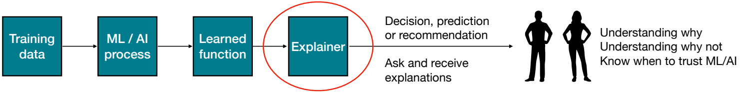
https://www.technologyreview.com/2018/10/10/139858/amazon-ditched-ai-recruitment-software-because-it-was-biased-against-women/

Explainability in the workflow



following (Gianfagna and Di Cecco, 2021)

Explainability in the workflow



following (Gianfagna and Di Cecco, 2021)

Interpretability, Explainability, causality...

- **Explainability**:
  - Human-agent interaction; explanatory agent revealing underlying causes to its or another agent's decision making (Miller, 2019).
  - Post-hoc interpretation; predictions without elucidating the mechanisms (Lipton, 2018).
- Distinguish explainability and interpretability, e.g. (Gianfagna and Di Cecco, 2021):
  - **Interpretability**: Understand (as a human) what the *model* does on known data.
  - **Explainability**: Broader understanding of the *model*, i.e., ability to understand counterfactuals.
- **Causality**: Understand causal effects in the *real world*.
- Not always possible to disentangle explainability and causality.

Examples from Finance

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

- **Buy-now-pay-later (BNPL)** systems make loan decisions in real time (Zalando, Klarna)
  $\rightsquigarrow$ unsecured loan to customer with no known credit history; revenue v fraud.
- **Risk management**, esp. stress-testing requires to know relationships between risk factors and portfolio value, e.g. (Packham and Woebbeking, 2019, 2023)
- **Model calibration** using deep neural networks in a real-time trading environment, e.g. (Brigo *et al.*, 2021; Yuan *et al.*, 2024).
- Examples, where explainability is less relevant:
  - A/B testing (???)
  - ???

Financial modelling: Why explainability?

- Observational data
- Noisy
- Heavy-tailed
- Jumps
- Non-stationary
- Autocorrelated

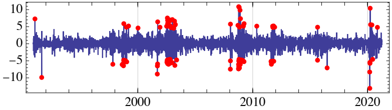
$$\frac{\mathrm{d}S_t}{S_{t-}} = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t - \lambda_t^+ \mathsf{E}(\mathbf{e}^{J^+} - 1)\,\mathrm{d}t - \lambda_t^- \mathsf{E}(\mathbf{e}^{J^-} - 1)\,\mathrm{d}t + (\mathbf{e}^{J^+} - 1)\,\mathrm{d}N_t^{(1)} + (\mathbf{e}^{J^-} - 1)\,\mathrm{d}N_t^{(2)}$$

$$\begin{pmatrix} \mathrm{d}\lambda_t^+ \\ \mathrm{d}\lambda_t^- \end{pmatrix} = \begin{pmatrix} \kappa^+(\theta^+ - \lambda_t^+) \\ \kappa^-(\theta^- - \lambda_t^-) \end{pmatrix} \mathrm{d}t + \begin{pmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{pmatrix} \begin{pmatrix} J^+\,\mathrm{d}N_t^{(1)} \\ J^-\,\mathrm{d}N_t^{(2)} \end{pmatrix},$$

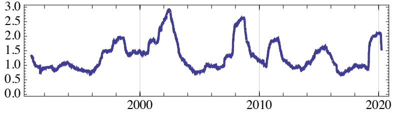
$$J^+ \sim \varpi^+, \; J^- \sim \varpi^-$$

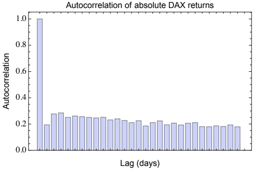
(Liu, Packham, Sepp, 2024)

DAX log returns



Moving average standard deviation (250 days) [%]





Regulatory aspects

- **EU-AI Act**: Categorizes risk from AI systems as minimal, limited, high, and unacceptable risk.
- **High-Risk AI** includes access to services (e.g. insurance, banking, credit, ...).
- Key requirements:
  - Transparency
  - Human oversight
  - Accuracy
  - Robustness

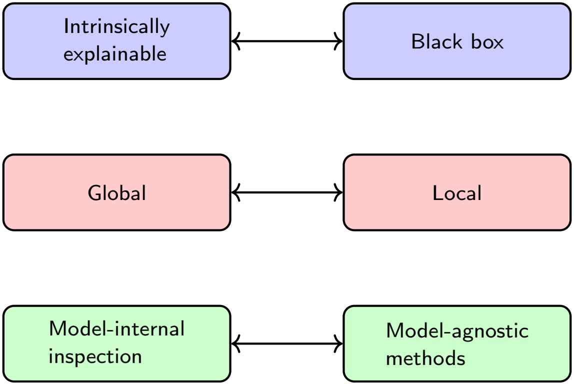
Regulatory aspects

- **EU-AI Act**: Categorizes risk from AI systems as minimal, limited, high, and unacceptable risk.
- **High-Risk AI** includes access to services (e.g. insurance, banking, credit, ...).
- Key requirements:
  - Transparency
  - Human oversight
  - Accuracy
  - Robustness

Do we really need explainability?

- **Model validation**: Does the model work as expected? Does the model manage extreme inputs? Is the model unbiased?
- **Model robustness**: Does the model work reasonably for unknown data? How does the model react to small changes in the inputs? Can we trust the model?
- **Knowledge discovery**: We are typically interested in more than prediction. What conclusions can we draw from model output?
- **"Human-in-the-loop"**: Collaboration of AI/ML and human may be much stronger than AI/ML alone.

Classification of XAI methods



Model-agnostic methods

- **Feature Importance**:
  - Measures how much each input feature contributes to the model's predictions.
  - Examples: **Permutation importance** (Breiman, 2001), **SHAP** (Shapley Additive Explanations) Lundberg and Lee (2017), **LIME** (Local Interpretable Model-Agnostic Explanations) (Ribeiro *et al.*, 2016).
- **Counterfactual Explanations**:
  - Provides "what-if" scenarios (e.g. what minimal change to the input would lead to a different output).
  - Useful for understanding decision boundaries and actionable insights.
- **Partial Dependence Plots (PDP)** (Friedman, 2001) and **Accumulated Local Effects (ALE)** (Apley and Zhu, 2020):
  - Show average effects of features.
- **Surrogate Models**:
  - Simplifies a complex model by approximating it with an interpretable model (e.g. decision trees, linear regression).

SHAP

- **Shapley values** (Shapley, 1953):
  - Concept in cooperative game theory
  - Method that allocates payouts (costs) to players depending on their contribution to the total payout (cost).
  - Players cooperate in a coalition and receive a certain profit from this cooperation.
- Example:
  - Alice wants to go to Adlershof; Bob wants to go to Britz; Charlie wants to go to Charlottenburg.
  - If they share a taxi that drops them off one at a time, what is a fair split-up of the total cost?
  - Shapley values: Marginal contributions of each person, averaged over all possible coalitions.

SHAP values

- **SHAP Values (Shapley Additive Explanations):** replace players by features, (Lundberg and Lee, 2017):

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left(f(S \cup \{i\}) - f(S)\right),$$

  where

  - ϕ_i: SHAP value for feature i
  - N: set of all features
  - S: Subset of all features N that does not include feature i
  - $|S|$: Number of features in subset S
  - $f(S)$: Model prediction based on the features in subset S

Example: Real-estate model

- Cooperation with **Scope Ratings GmbH** (and HTW Berlin) on valuing and risk management of real estate portfolios using publicly available data
- Approx. 1.24m transactions in Île-de-France, 2014-2022
- Open Street Map (OSM) data
- Macroeconomic data
- XGBoost (eXtreme Gradient Boosting)

SHAP values XGB

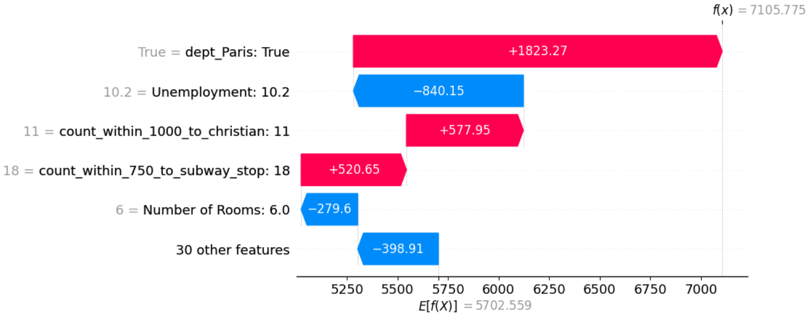


Figure: SHAP waterfall plot for a random apartment in Paris

SHAP values XGB

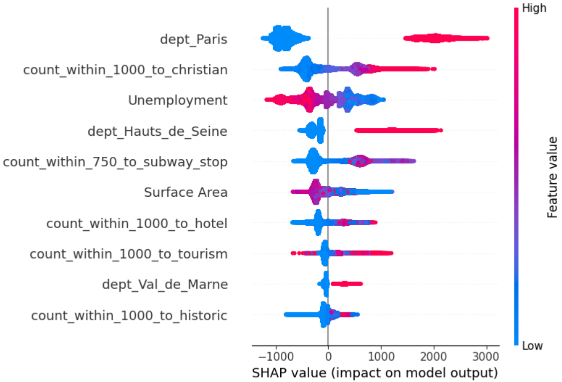
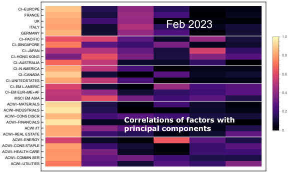
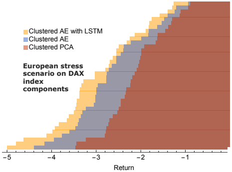


Figure: SHAP summary plot

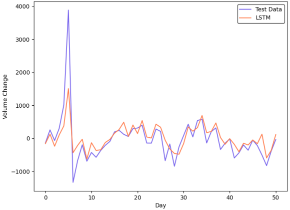
Further applications

- **Stress-testing, reverse stress testing**, (Packham, 2024):
  - Dimension reduction and aggregation of risk factors.
  - Explainable PCA and explainable auto-encoder.





- **Fractional trading**:
  - Cooperation with **Upvest GmbH** and HTW Berlin
  - Risk Management, esp. hedging of fractions
  - Drivers: News sentiment, trading volume, ...



Outlook

- Significance (in statistical sense) when data are dependent, see e.g. (Aas *et al.*, 2021).
- Interpretation of coefficients as in linear regression: Double machine learning, (Chernozhukov *et al.*, 2018)

References

Aas, K., M. Jullum, and A. Løland. Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *Artificial Intelligence*, 298:103502, September 2021.

Angrist, J. D. and J.-S. Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

Apley, D. W. and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, June 2020.

Aronson, D. R. *Evidence-Based Technical Analysis*. Wiley & Sons, 2006.

Bernard, E. *Introduction to Machine Learning*. Wolfram Media, 2021.

Bourlard, H. and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, sep 1988.

Breiman, L. Random forests. *Machine learning*, 45:5–32, 2001.

Brigo, D., X. Huang, A. Pallavicini, and H. S. d. O. Borde. Interpretability in deep learning for finance: a case study for the heston model, 2021.

Büchel, P., M. Kratochwil, M. Nagl, and D. Rösch. Deep calibration of financial models: turning theory into practice. *Review of Derivatives Research*, 25(2):109–136, August 2021.

Buehler, H., L. Gonon, J. Teichmann, and B. Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, February 2019.

References

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018.

Cunningham, S. *Causal Inference – The Mixtape*. Yale University Press, 2021.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, December 1989.

De Spiegeleer, J., D. B. Madan, S. Reyners, and W. Schoutens. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, July 2018.

Dumitrescu, E., S. Hué, C. Hurlin, and S. Tokpavi. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European Journal of Operational Research*, 297(3):1178–1192, March 2022.

Fama, E. F. and K. R. French. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33(1):3–56, 1993.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Funahashi, H. Deep learning for derivatives pricing: a comparative study of asymptotic and quasi-process corrections. *Annals of Operations Research*, July 2024.

Gianfagna, L. and A. Di Cecco. *Explainable AI with Python*. Springer, 2021.

References

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Han, J., Y. Huang, S. Liu, and K. Towey. Artificial intelligence for anti-money laundering: a review and extension. *Digital Finance*, 2(3-4):211–239, 2020.

Hinton, G. E. and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, jul 2006.

Hornik, K., M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Hull, J. *Machine Learning in Business*. Independently published, 3rd edition, 2021.

Imbens, G. W. and D. B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, April 2015.

Jacod, J. and P. Protter. *Probability Essentials*. Springer, 2nd edition, 2003.

James, G., D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Jensen, R. I. T. and A. Iosifidis. Fighting money laundering with statistics and machine learning. *IEEE Access*, 11:8889–8903, 2023.

Kallenberg, O. *Foundations of modern probability*. Springer Science & Business Media, 2nd edition, 2002.

Koop, G. *Introduction to Econometrics*. Wiley, 2008.

References

Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, feb 1991.

Lapuschkin, S., S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1), 2019.

Larsen, R. J., M. L. Marx, et al. *An introduction to mathematical statistics and its applications*. Prentice-Hall Englewood Cliffs, NJ, 5th edition, 2012.

Lipton, Z. C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, June 2018.

Liu, F., N. Packham, and A. Sepp. Jump risk premia in the presence of clustered jumps. Working paper, 2024.

Lundberg, S. M. and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4768–4777, 2017.

Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, February 2019.

Ni, H., X. Dong, J. Zheng, and G. Yu. *An Introduction to Machine Learning in Quantitative Finance*. World Scientific, 2021.

Packham, N. and C. F. Woebbeking. A factor-model approach for correlation scenarios and correlation stress testing. *Journal of Banking & Finance*, 101:92–103, 2019.

References

Packham, N. and F. Woebbeking. Correlation scenarios and correlation stress testing. *Journal of Economic Behavior & Organization*, 205:55–67, 2023.

Packham, N. Risk factor aggregation and stress testing. *Quantitative Finance*, 24(9):1327–1340, 2024.

Ribeiro, M. T., S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144. ACM, 2016.

Rosenblatt, M. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

Ruf, J. and W. Wang. Neural networks for option pricing and hedging: a literature review. *The Journal of Computational Finance*, 2020.

Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, 2, 1953.

Shiryaev, A. N. *Probability*. Springer, Berlin, 2nd edition, 1996.

Torno, A., D. R. Metzler, and V. Torno. Robo-what?, robo-why?, robo-how? –a systematic literature review of robo-advice. In *Proc. Pacific Asia Conference on Information Systems*, 2021.

West, K. D. Asymptotic inference about predictive ability. *Econometrica*, pages 1067–1084, 1996.

White, H. A reality check for data snooping. *Econometrica*, pages 1097–1126, 2000.

Yuan, B., D. Brigo, A. Jacquier, and N. Pede. Deep learning interpretability for rough volatility, 2024.

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Prof. Dr. Natalie Packham
Professor of Mathematics and Statistics
Berlin School of Economics and Law
Department of Business and Economics
Badensche Str. 52
10825 Berlin

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law