esade

Intermediate Python & a Glimpse into AI applications

Class 5

Pepe Bonet Giner

17th January 2024

Index Class 5

Topic 1: Short recap

Topic 2: Intro to ML models in Python. Linear Regression

Topic 3: Scitkit Learn models Overview

esade

Topic 1: Recap

Recap: How do we learn from the past? We need something. **A model**

Identifier | Features | Label

Data From the past

ID	#Bedrooms	#bathrooms	M²	Price(Euro)
1	3	2	120	190.000
2	3	3	150	250.000
3	1	1	60	100.000
4	2	2	80	140.000

Train a model using the data from the past

Predictive Model
Black box

Trained Model

New data point:
ID: 1234
#Bedrooms: 5
#bathrooms: 4
M² : 200

350.000

Use the trained model to predict the price of future/ new data point

Recap: What is a model?

- Models are programs that can find patterns in the data that is fed to them

#Bedrooms	#bathrooms	M²
3	2	120
3	3	150
1	1	60
2	2	80

Price(Euro)
190.000
250.000
100.000
140.000

More bathrooms more money
More square meter more money
Less noise more money
Less bedrooms less money…

AI/ML. Classification & regression

ML: Data (input) + Answers (predictive label) —> Rules (Model)

Input Data

#Bedrooms	#bathrooms	M²
3	2	120
3	3	150
1	1	60
2	2	80

+

Answers

Price(Euro)
190.000
250.000
100.000
140.000

=

Rules

Model

Fault
No
Yes
No
No

Classification

Price(Euro)
190.000
250.000
100.000
140.000

Regression

Classification is about predicting a label (Categorical) and regression is about predicting a quantity (Numerical)

Computers only understand numbers!

Categorical Features

Numerical Features

Gender	Parent Education	Age	Grade
Male	College	21	8
Female	High School	22	9
Male	None	20	6
Female	College	21	5

Computers need to code the categorical features/ columns as numbers. How do we do it?

We have two main approaches:
- Label Encoding
- One Hot Encoding

Label Encoding

Parent Education
College
High School
None
College

Parent Education
2
1
0
2

One Hot Encoding

Parent Ed_College	Parent Ed_High School	Parent Ed_None
1	0	0
0	1	0
0	0	1
1	0	0

Data Normalisation Practice

$$x_{inorm} = \frac{x_i - X_{min}}{X_{max} - X_{min}} \cdot (range_{max} - range_{min}) + range_{min}$$

Age
10
25
53
42

Age_norm
0
0.35
1
0.74

- Write a function that normalizes any column that you input in the range [0, 1] and test it.
- Check that your output is correct using MinMaxScaler from scikit learn

What are we going to predict?

- The math score will be our objective. The column we are going to predict.

What we should all have?

```
In [13]: data.columns
Out[13]: Index(['math score', 'reading score', 'writing score', 'id_student', 'Year',
                'Age', 'gender_male', 'race/ethnicity_group B',
                'race/ethnicity_group C', 'race/ethnicity_group D',
                'race/ethnicity_group E',
                'parental level of education_bachelor's degree',
                'parental level of education_high school',
                'parental level of education_master's degree',
                'parental level of education_some college', 'lunch_standard',
                'test preparation course_none'],
               dtype='object')
```

```
In [19]: data.head()
Out[19]:
```

	math score	reading score	writing score	id_student	Year	Age	gender_male	race/ethnicity_group B	race/
0	67	0.611765	0.519481	1000	2023	14.0	1	0	
1	40	0.164706	0.415584	1001	2023	17.0	0	0	
2	59	0.529412	0.350649	1002	2023	14.0	1	0	
3	77	0.741176	0.584416	1003	2023	17.0	1	1	
4	78	0.682353	0.584416	1004	2023	16.0	1	0	

esade

Topic 2: Intro to ML models in Python. Linear Regression

Train & Test sets. Why are they necessary?

Training Set

Gender	Parent Education	Age	Grade
Male	College	21	8
Female	High School	22	9

To train the model. Can represent up to 90 % of the data

Test Set

Gender	Parent Education	Age	Grade
Male	None	20	6
Female	College	21	5

To test how the model generalizes to never seen data

Train & Test sets. Why are they necessary?

Training Set

Gender	Parent Education	Age	Grade
Male	College	21	8
Female	High School	22	9

To train the model. Can represent up to 90 % of the data

Test Set

Gender	Parent Education	Age	Grade
Male	None	20	6
Female	College	21	5

To test how the model generalizes to never seen data

Overfitting: The model is not good on never seen data

Underfitting: The model is bad everywhere

The reason to have a separate train and test set is to avoid overfitting of the model to the data

Exercise

- Generate a train and test set of your dataset (Hint: Use a sci-kit learn function)

Our Pandas for ML models is Scikit Learn

Scikit learn is an ML library for Python that features various classification, regression and clustering algorithms

It will allow us to build a model and make predictions in 4 lines of code

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

```
# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(trainX, trainY)

# Make predictions using the testing set
y_pred = regr.predict(testX)

# The mean absolute error
print("MAE: %.2f" % mean_absolute_error(testY, y_pred))
```

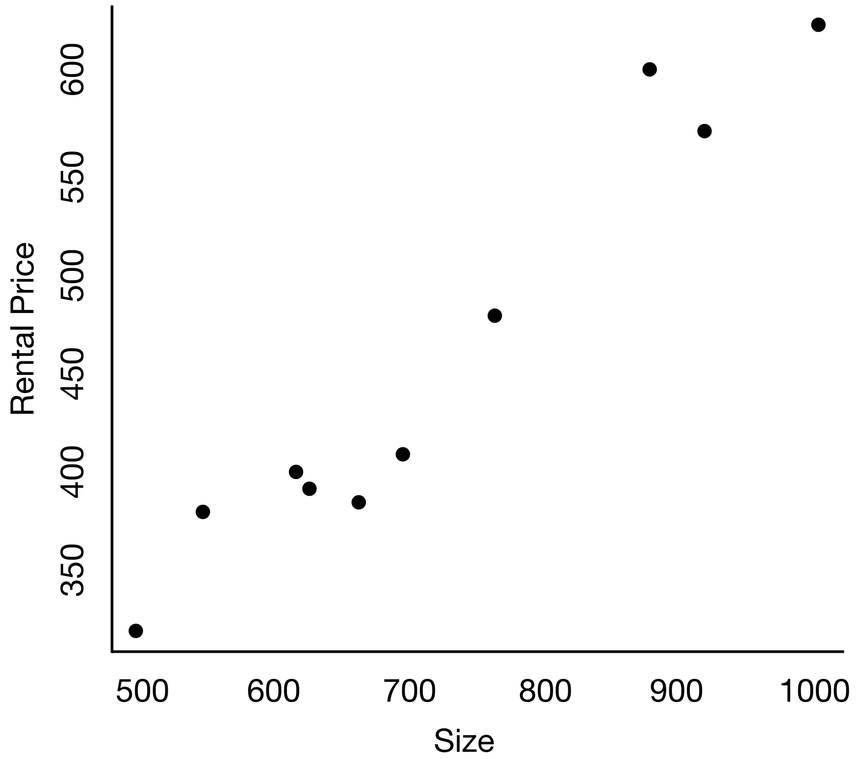
```
MAE: 4.41
```

Linear regression. Explained

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

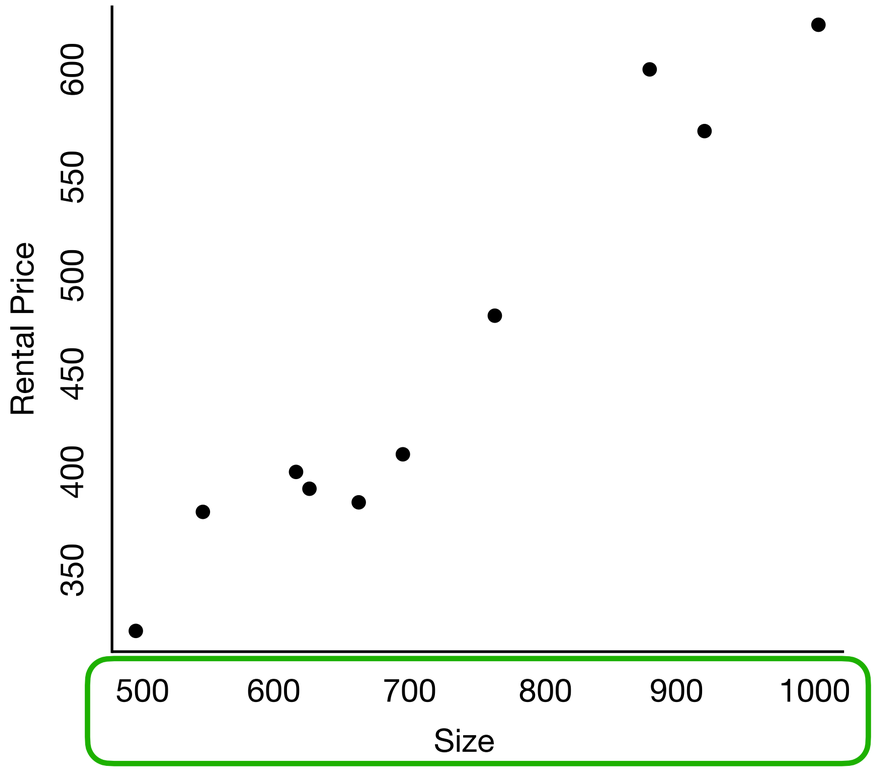
Linear regression. Explained

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620



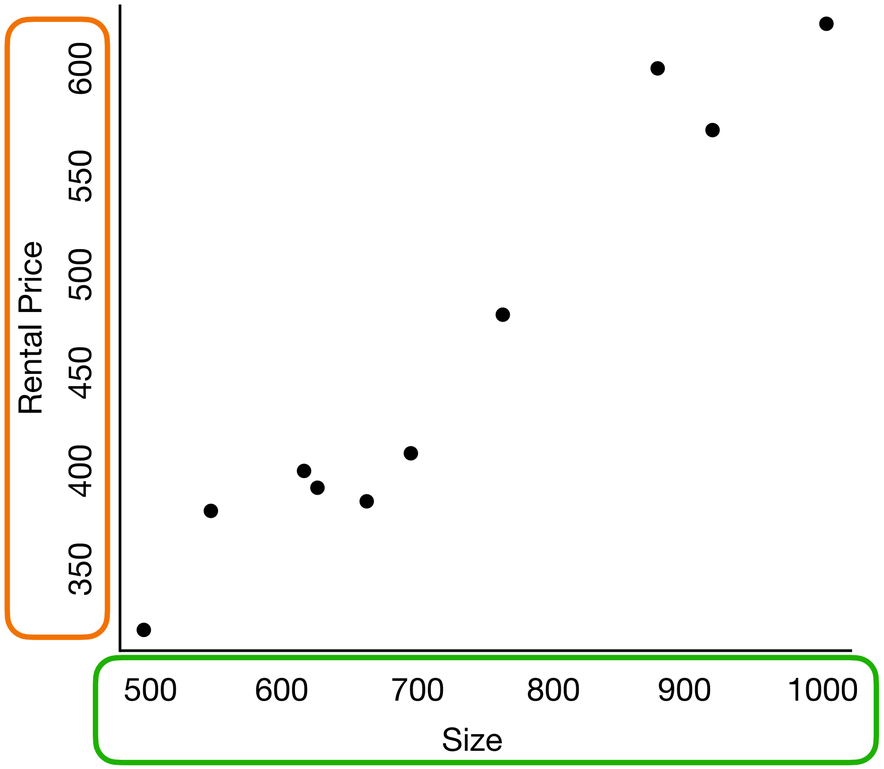
Linear regression. Explained

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620



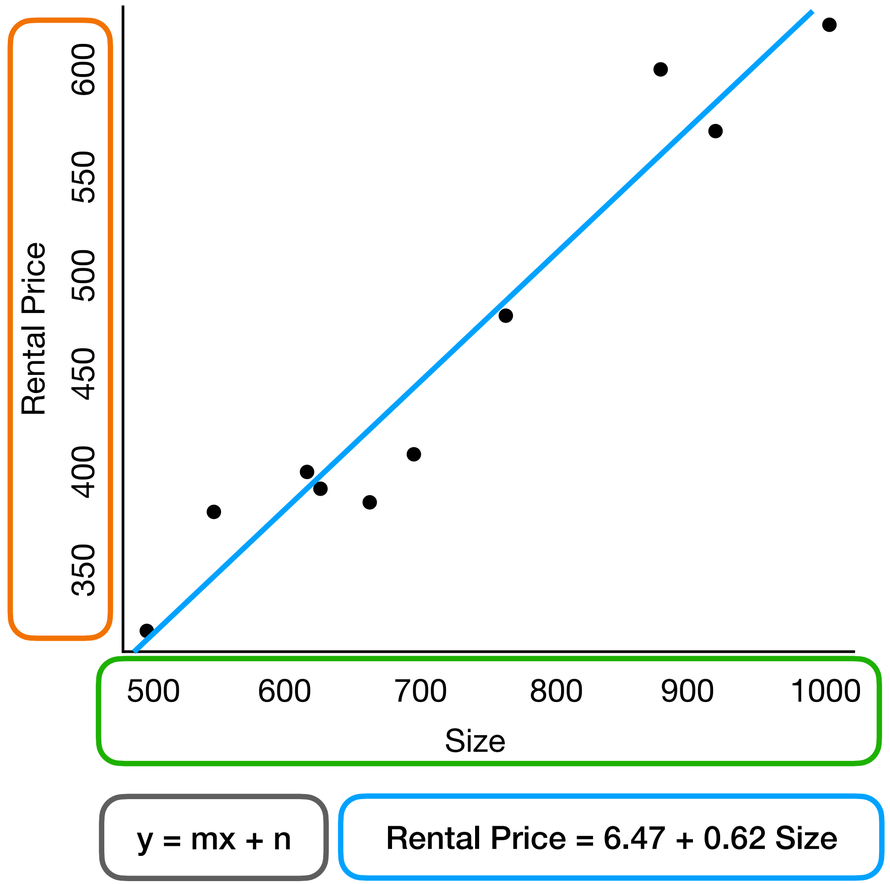
Linear regression. Explained

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620



Linear regression. Explained

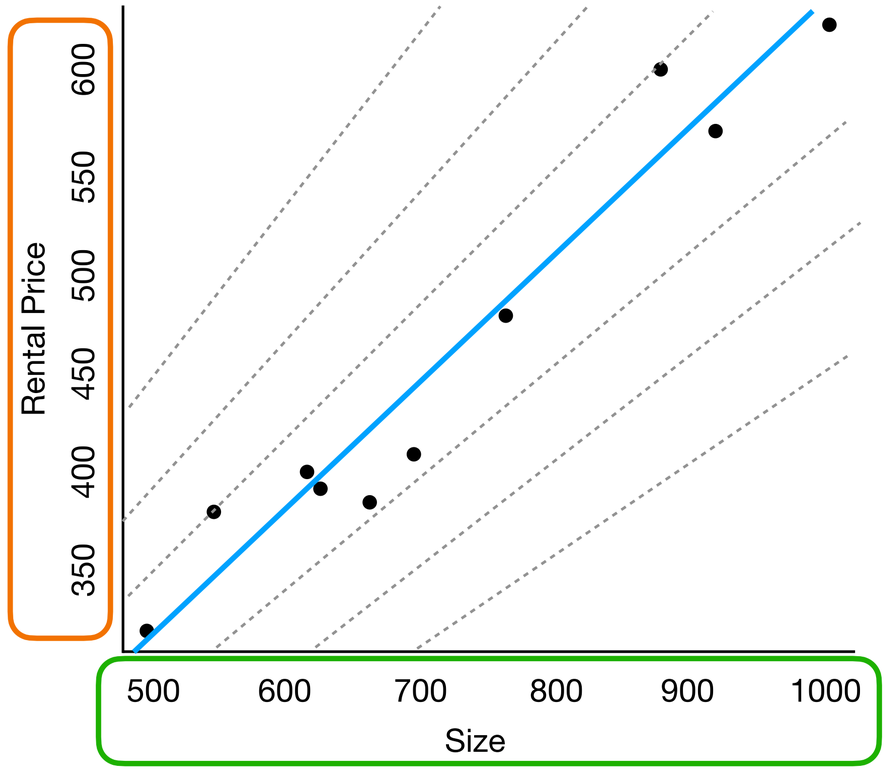
ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620



Linear regression. Explained

How do we choose a line? Which one fits better?

ID	Size	Floor	Broadband Rate	Energy Rating	Rental Price
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

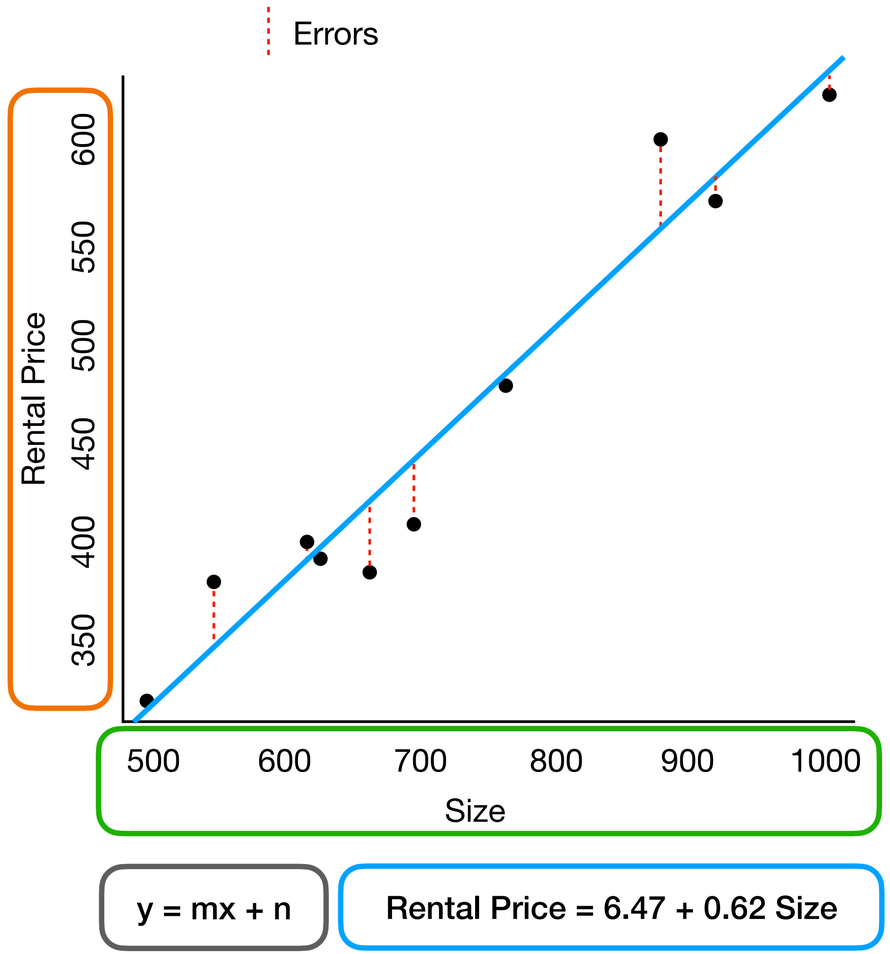


$y = mx + n$

Rental Price = 6.47 + 0.62 Size

Measuring the error. Sum of Squared Errors

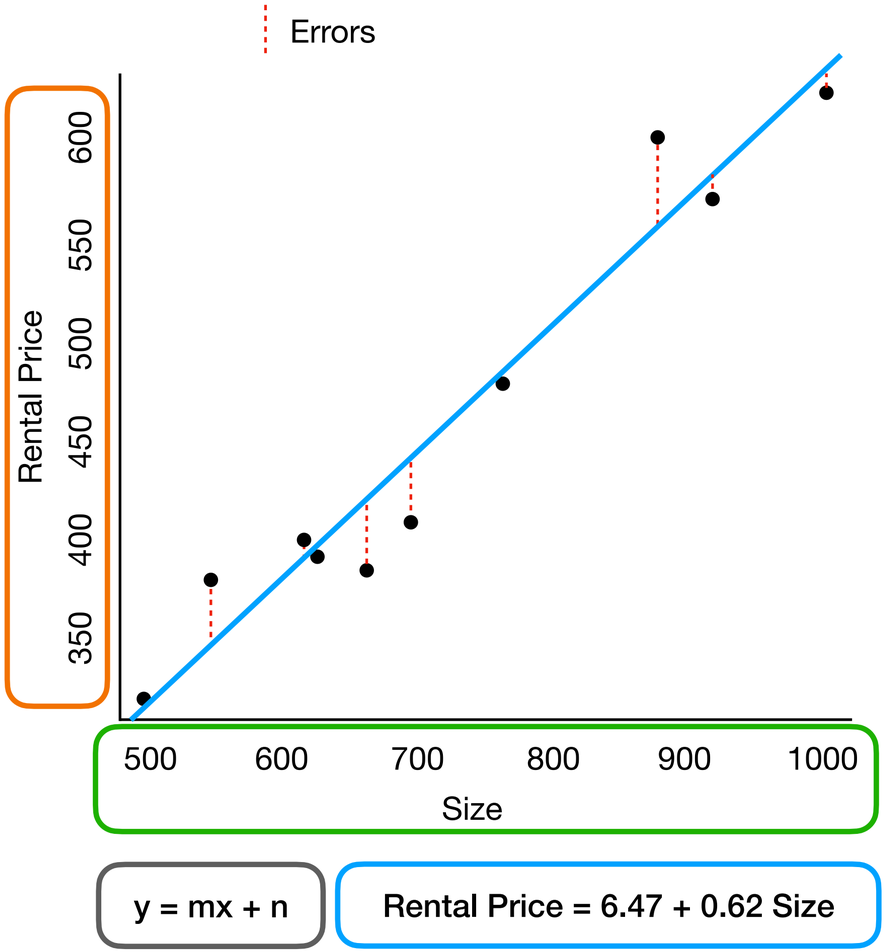
We need an **error function** that captures the errors between the predictions of the model (blue line) and the actual values in the dataset (Real rental price)



Measuring the error. Sum of Squared Errors

We need an **error function** that captures the errors between the predictions of the model (blue line) and the actual values in the dataset (Real rental price)

Bad Computer

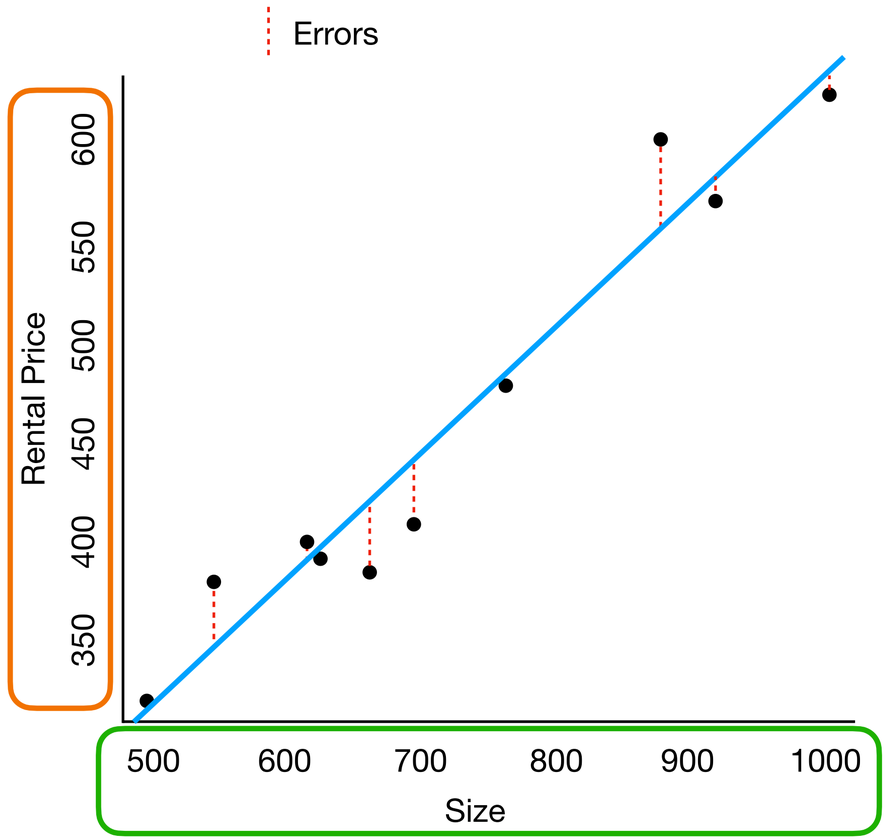


Measuring the error. Sum of Squared Errors

We need an **error function** that captures the errors between the predictions of the model (blue line) and the actual values in the dataset (Real rental price)

Sum of Squared Errors

$$L_2 = \frac{1}{2}\sum_{i=1}^{n}(y - y_{pred})^2$$



y = mx + n

Rental Price = 6.47 + 0.62 Size

Measuring the error. Sum of Squared Errors

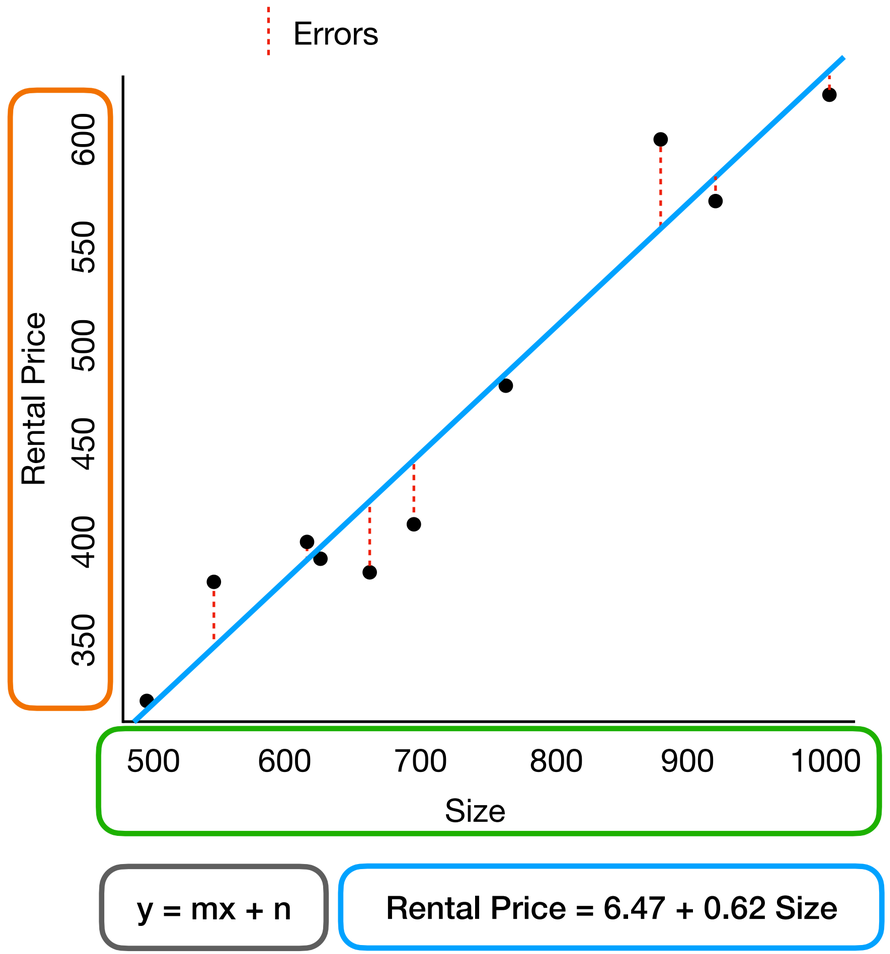
We need an **error function** that captures the errors between the predictions of the model (blue line) and the actual values in the dataset (Real rental price)

Sum of Squared Errors

$$L_2 = \frac{1}{2}\sum_{i=1}^{n}(y - y_{pred})^2$$

$$L_2 = \frac{1}{2}\sum_{i=1}^{n}(y - (mx + n))^2$$

$$L_2 = \frac{1}{2}\sum_{i=1}^{n}(y - (0.62 \cdot Size + 6.47))^2$$

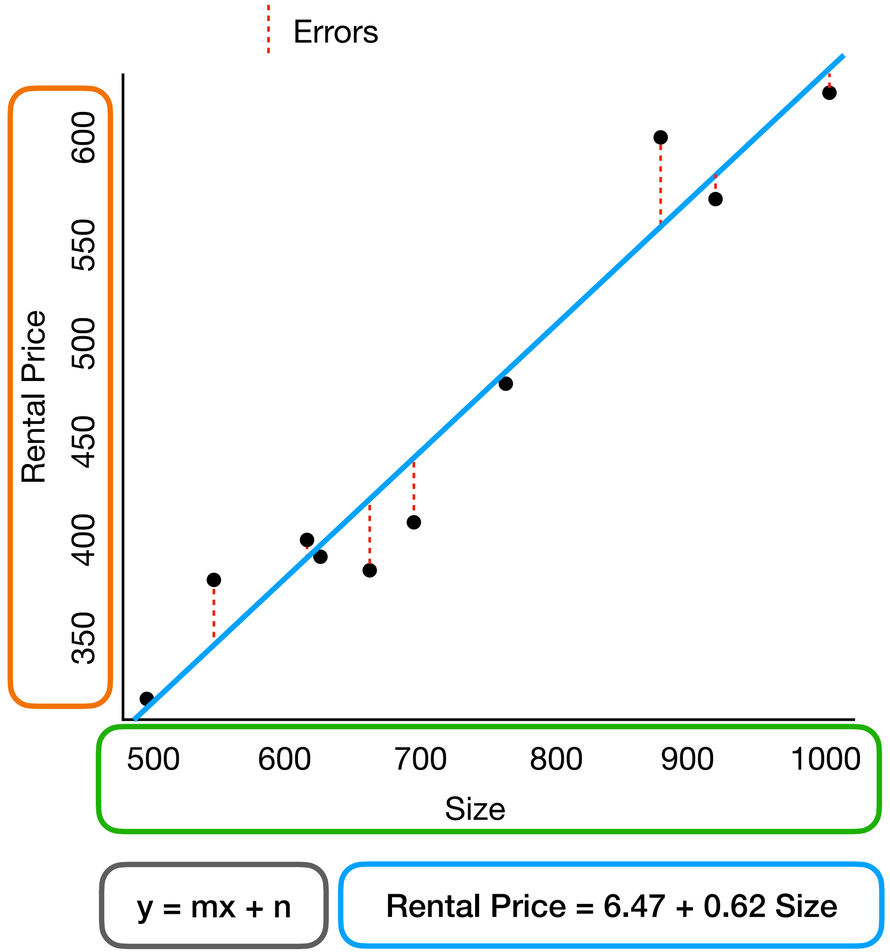


Objective: Find the best fit of a line to the data

We want to obtain the line that minimizes the sum of squared errors.

$$min(L_2)$$

To find the best parameters for the line that we are searching for (m and n in this case)



Objective: Find the best fit of a line to the data

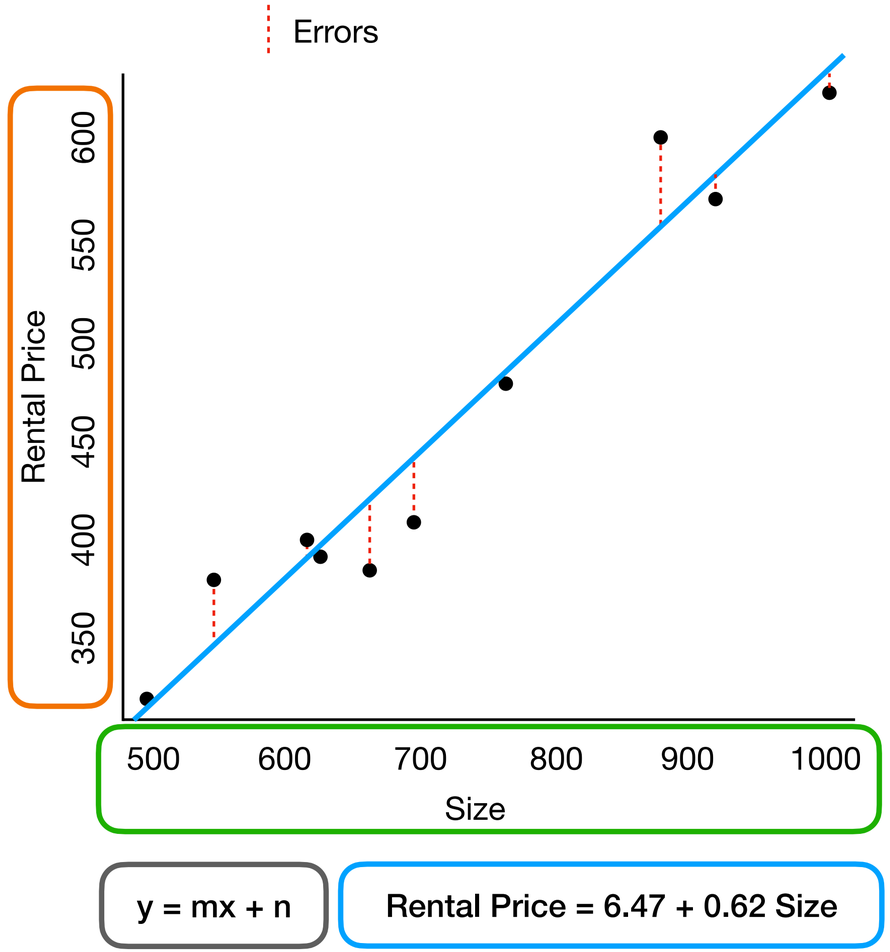
We want to obtain the line that minimizes the sum of squared errors.

$$min(L_2)$$

To find the best parameters for the line that we are searching for (m and n in this case)

This search can be done through something called **gradient descent**

This method is at the core of many machine learning models



Linear regression model in Python to predict math scores

The technical aspects explained and many more are already handled by sci-kit learn when building any model

Therefore, our work consists of having the data ready and selecting the model. After that we are done in some lines of code

Load packages

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

Linear regression model in Python to predict math scores

The technical aspects explained and many more are already handled by sci-kit learn when building any model

Therefore, our work consists of having the data ready and selecting the model. After that we are done in some lines of code

Load packages

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

Start linear model

```
# Create linear regression object
regr = linear_model.LinearRegression()
```

Linear regression model in Python to predict math scores

The technical aspects explained and many more are already handled by sci-kit learn when building any model

Therefore, our work consists of having the data ready and selecting the model. After that we are done in some lines of code

Load packages

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

Start linear model

```
# Create linear regression object
regr = linear_model.LinearRegression()
```

Fit Model to data

```
# Train the model using the training sets
regr.fit(trainX, trainY)
```

Linear regression model in Python to predict math scores

The technical aspects explained and many more are already handled by sci-kit learn when building any model

Therefore, our work consists of having the data ready and selecting the model. After that we are done in some lines of code

Load packages

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

Start linear model

```
# Create linear regression object
regr = linear_model.LinearRegression()
```

Fit Model to data

```
# Train the model using the training sets
regr.fit(trainX, trainY)
```

Test your model

```
# Make predictions using the testing set
y_pred = regr.predict(testX)
```

Linear regression model in Python to predict math scores

The technical aspects explained and many more are already handled by sci-kit learn when building any model

Therefore, our work consists of having the data ready and selecting the model. After that we are done in some lines of code

Load packages

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_absolute_error
```

Start linear model

```
# Create linear regression object
regr = linear_model.LinearRegression()
```

Fit Model to data

```
# Train the model using the training sets
regr.fit(trainX, trainY)
```

Test your model

```
# Make predictions using the testing set
y_pred = regr.predict(testX)
```

Performance measurement

```
# The mean absolute error
print("MAE: %.2f" % mean_absolute_error(testY, y_pred))
```

```
MAE: 4.41
```

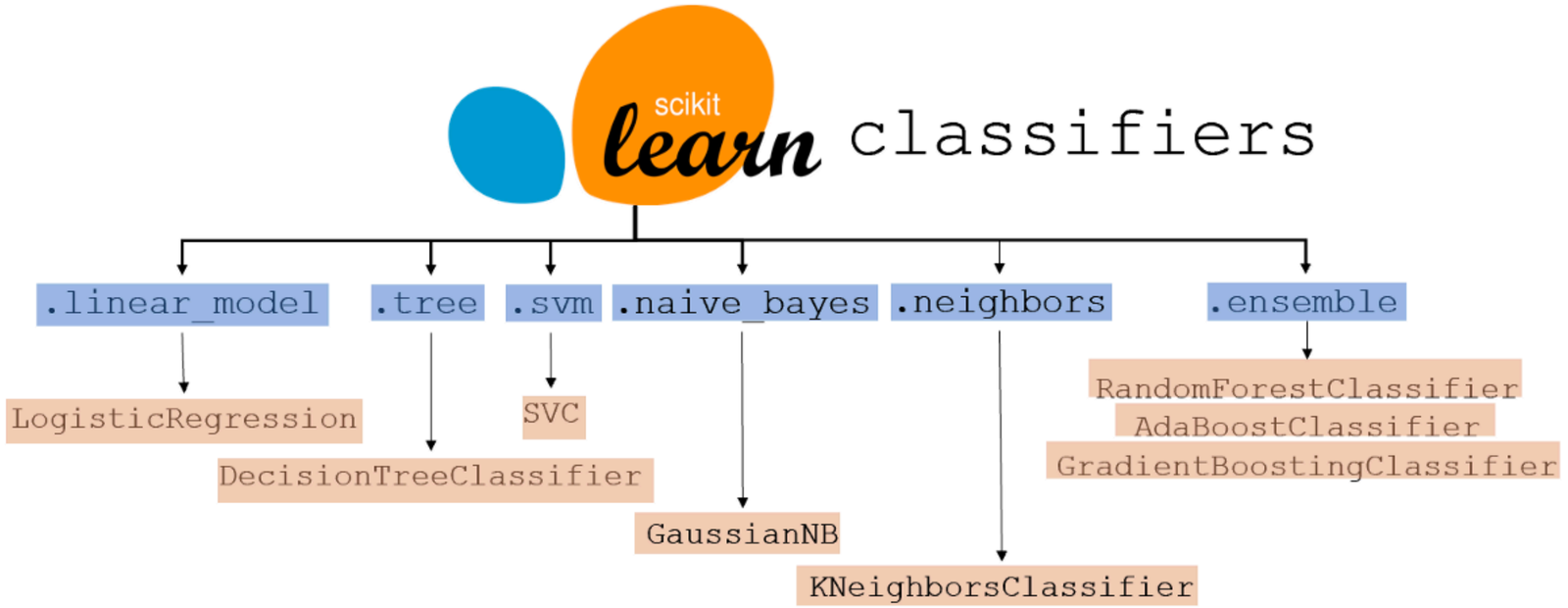
Exercise

- Code your first Linear regression model and get how good it does

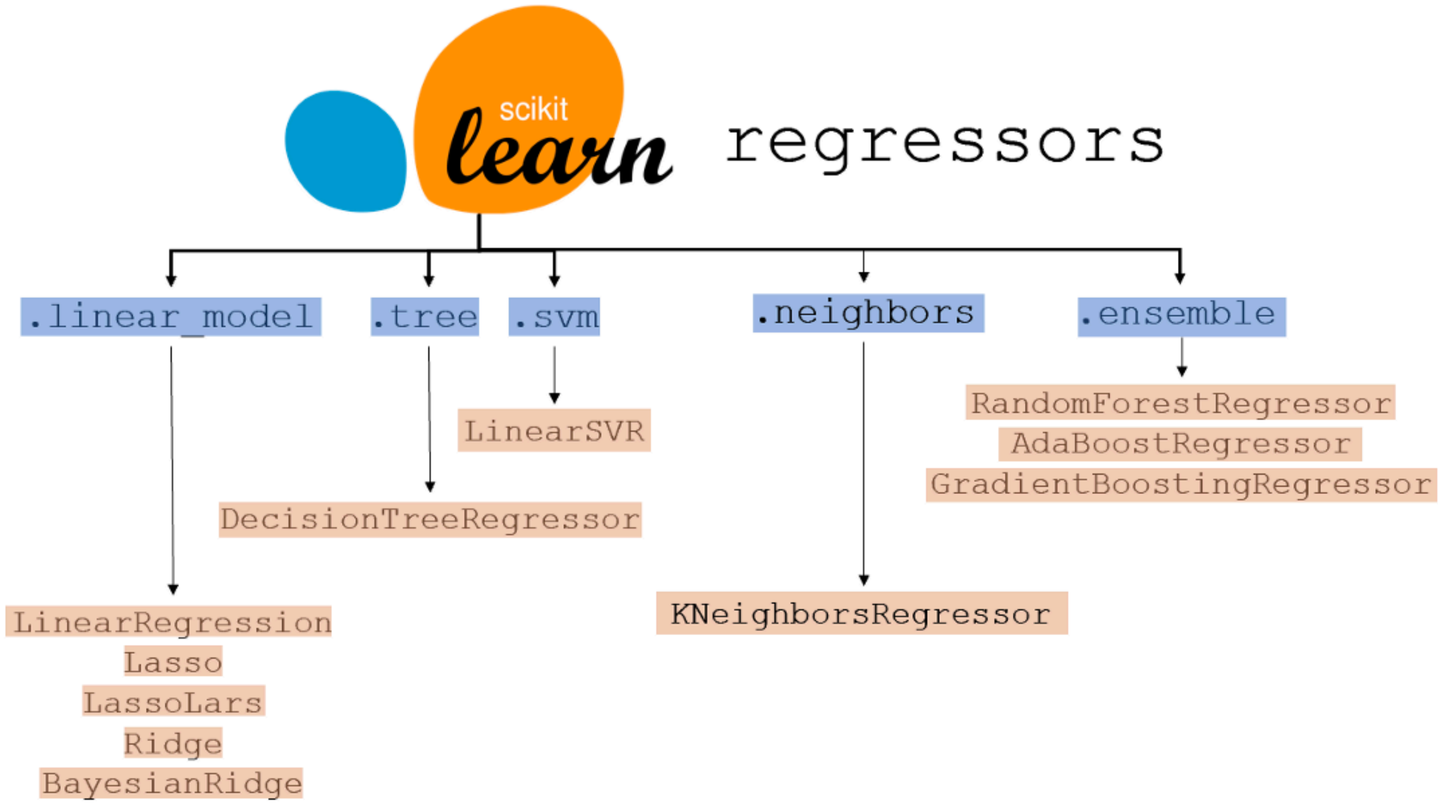
esade

Topic 3: Scitkit Learn models Overview and More

Scikit learn models for Classification problems

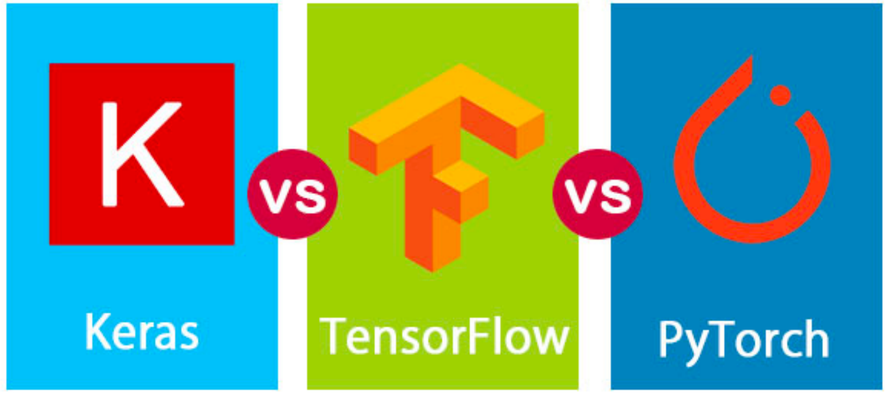


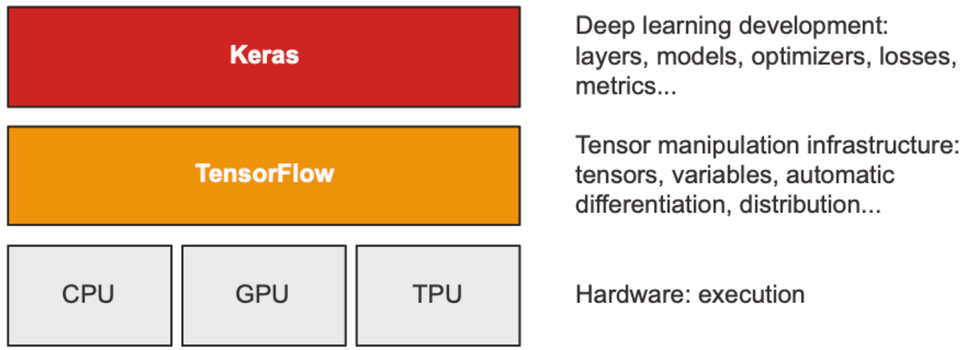
Scikit learn models for Regression problems



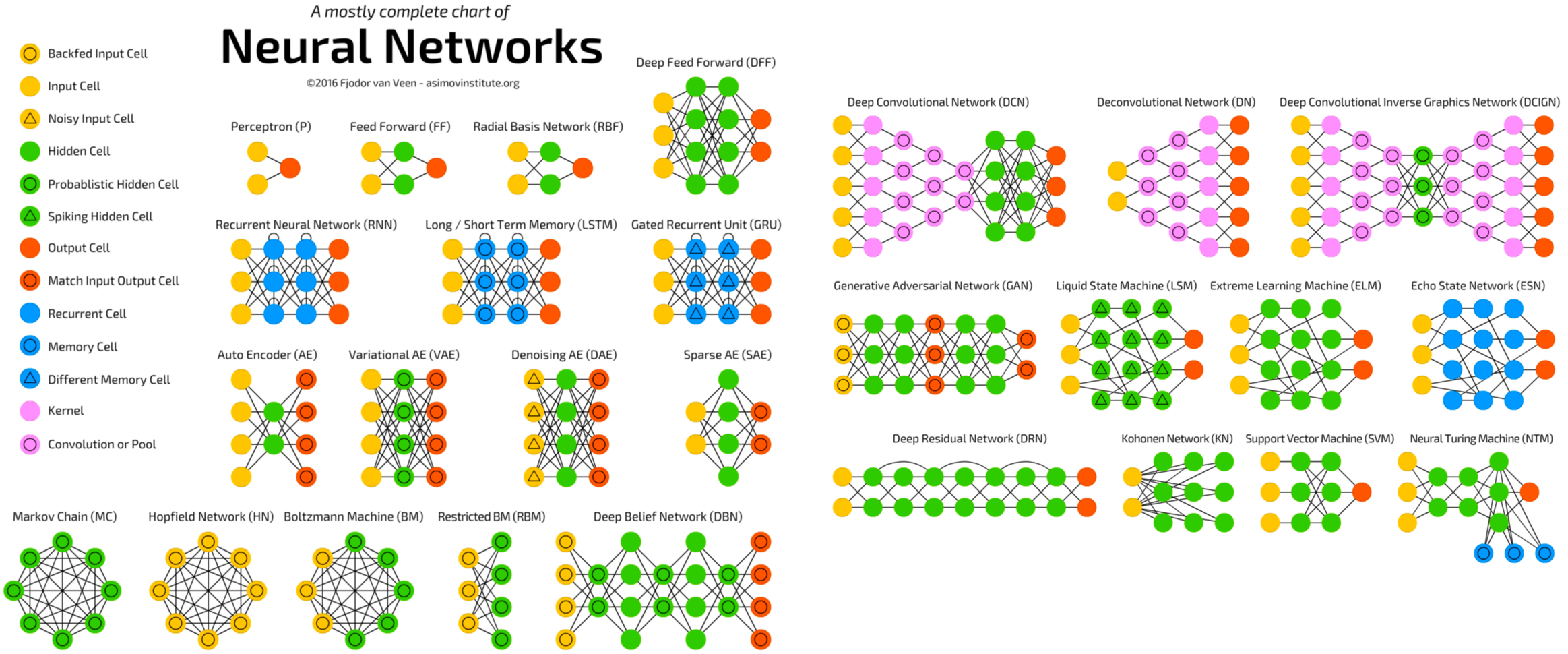
You can also “easily” build Neural Networks in Python


```
>>> from sklearn.datasets import make_regression
>>> from sklearn.ensemble import GradientBoostingRegressor
>>> from sklearn.model_selection import train_test_split
>>> X, y = make_regression(random_state=0)
>>> X_train, X_test, y_train, y_test = train_test_split(
...     X, y, random_state=0)
>>> reg = GradientBoostingRegressor(random_state=0)
>>> reg.fit(X_train, y_train)
GradientBoostingRegressor(random_state=0)
>>> reg.predict(X_test[1:2])
array([-61...])
>>> reg.score(X_test, y_test)
0.4...
```





And for that you also have many options



Every model is a bit of a different world

- Some do not need normalization some others do
- Some need to take care of correlated variables some others don’t
- Some are better than others when fewer data is available
- Some are better for certain tasks and problems
- The math behind it is different for the different models; therefore, the way they learn also differs (Maximum Likelihood (Probability-based), Gradient-based methods…)

Every model is a bit of a different world

- Some do not need
- Some need to take
- Some are better th
- Some are better fo
- The math behind i
  learn also differs (
  methods…)

rs don’t

erefore, the way they

, Gradient-based

Exercise

- Does anybody see a problem with the time at which we did the normalization step?

Exercise

- Does anybody see a problem with the time at which we did the normalization step?
- The train test split should come first and then normalize each group separately. Example of data leakage. Try fixing this in your code.

esade

Documentation

Datacamp Courses

Python Fundamentals

Are you ready to gain the foundational skills you need to become a Python programmer? In this track, you'll learn the Python basics you need to start on your programming journey, including how to clean real-world data ready for analysis, use data visualization libraries, and even how to write your own Python functions.

Your instructor Hugo will introduce you to how companies worldwide use Python to gain a competitive edge. Through hands-on coding exercises you'll then learn how to store, manipulate, and explore data using NumPy. Then it's time to level-up as you learn how to visualize your data using Matplotlib, manipulate DataFrames and dictionaries using pandas, and write your own functions and list comprehension. Start this track to add these essential Python skills to your data science toolbox.

Switch Track

Python 15 hours 4 Courses 1 Skill Assessment

Data Scientist with Python

Gain the career-building Python skills you need to succeed as a data scientist. No prior coding experience required.

In this track, you'll learn how this versatile language allows you to import, clean, manipulate, and visualize data—all integral skills for any aspiring data professional or researcher. Through interactive exercises, you'll get hands-on with some of the most popular Python libraries, including pandas, NumPy, Matplotlib, and many more. You'll then work with real-world datasets to learn the statistical and machine learning techniques you need to train decision trees and use natural language processing (NLP). Start this track, grow your Python skills, and begin your journey to becoming a confident data scientist.

Resume Track

Python 88 hours 23 Courses 6 Projects 3 Skill Assessments

Free Books to learn Python & data analytics