DATA SCIENCE 2 - DATA & A.I.

3 V MACHINE LEARNING

4-7 ML TECHNIQUES

Karel de Grote Hogeschool

DATA SCIENCE 2
DATA & A.I. 3

I

PYTHON BASICS

Python for data science

II

WORKING WITH ARRAYS

Numpy

III

DATA ENGINEERING

pandas

IV

DATA VISUALISATION

Matplotlib

V

MACHINE LEARNING

Automatically find patterns

V

MACHINE LEARNING

scikit-learn

MACHINE LEARNING

01 WHAT IS MACHINE LEARNING
Automatically find patterns

02 INTRODUCING SCIKIT-LEARN
Machine learning with Python

03 HYPERPARAMETERS AND CROSS VALIDATION
Holdout samples and cross-validation

04 REGRESSION
Best fitting line

05 DECISION TREES
Best separating lines

06 K-MEANS CLUSTERING
Object grouping

07 ASSOCIATION RULES
Frequent itemsets

08 ARTIFICIAL NEURAL NETWORK
Imitate the human brain

SUPERVISED LEARNING: REGRESSION

Linear and Polynomial Regression

4

REGRESSION
(SUPERVISED LEARNING)

What?

prediction of numerical outcomes

How?

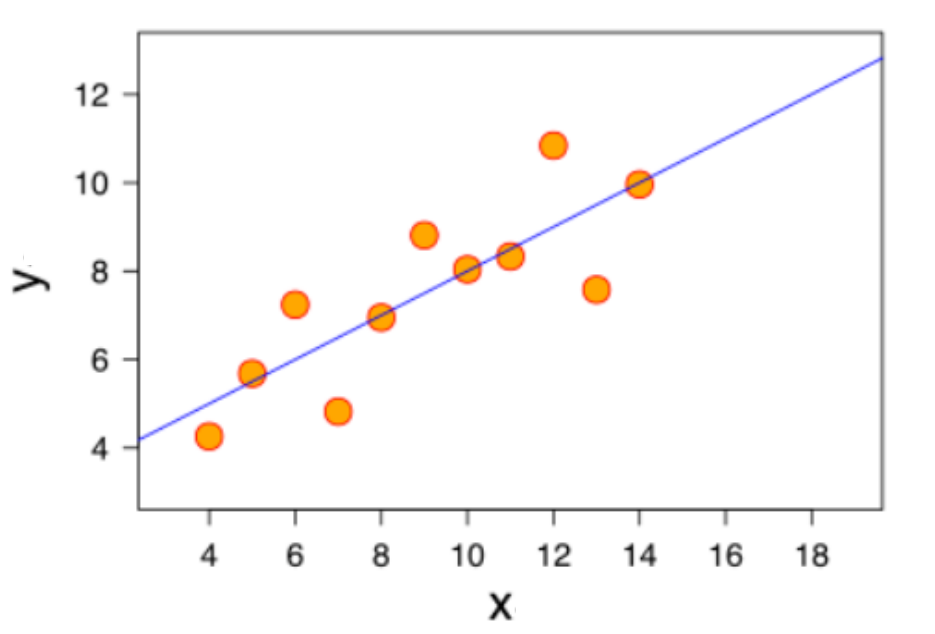
Fit a mathematical function to the data to model the relationship between features and the target.

Linear Regression: Fits a linear function to predict a numerical outcome.
Polynomial Regression: Fits a polynomial function (with varying degrees) to capture non-linear relationships in the data.

LINEAR REGRESSION

Prediction of numerical features

- One predictor (independent feature):
  Try to fit a line to the data in the best possible way: **$y = ax + b$**
  a: slope
  b: intercept



- Multiple predictors: **$y = b + a_1x_1 + a_2x_2 + \ldots + a_nx_n$**
  n: number of independent features
  $a_1 \ldots a_n$: slopes
  b: intercept

LINEAR REGRESSION WITH SCIKIT-LEARN

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
```

Hyperparameters:

fit_intercept

- **Type**: boolean, default = True
- **Description**: Determines whether or not the model should calculate the intercept (also called the bias term).If True, the model calculates the intercept. If False, the model assumes the data is already centered around the origin (i.e., the intercept is 0).
- **Usage**: fit_intercept=False may be useful if you know the data is already centered or you want a model without an intercept.

copy_X

- **Type**: boolean, default = True
- **Description**: If True, X will be copied before fitting the model. If False, changes to the original data (such as scaling) will affect the original dataset. It's usually safer to leave this as True.

n_jobs

- **Type**: int, default = None (means 1)
- **Description**: This specifies the number of CPU cores to use when fitting the model. If n_jobs=-1, all available cores will be used. This can help speed up the model fitting when working with large datasets.

LINEAR REGRESSION WITH SCIKIT-LEARN

One predictor

```
# DATA PREPARATION
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split

iris = sns.load_dataset('iris')

# Predictor: Pandas DataFrame, not Series!
X = iris[['petal_width']]
# Target feature to predict: Pandas Series
y = iris['petal_length']

# SPLIT in TRAIN and TEST dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

LINEAR REGRESSION WITH SCIKIT-LEARN

```
# MODEL SELECTION AND HYPERPARAMETER SELECTION (MODEL SPECIFIC)
from sklearn.linear_model import LinearRegression
model = LinearRegression()

# List all selected hyperparameters
print(model.get_params(deep=True))

# DERIVE MODEL FROM LABELED DATA (TRAIN MODEL/FIT MODEL)
model.fit(X_train, y_train)

# DISPLAY COEFFICIENTS (intercept and slope)
print(f"Intercept: {model.intercept_}")
print(f"Coefficient: {model.coef_[0]}")  # only 1 slope here
```

LINEAR REGRESSION WITH SCIKIT-LEARN

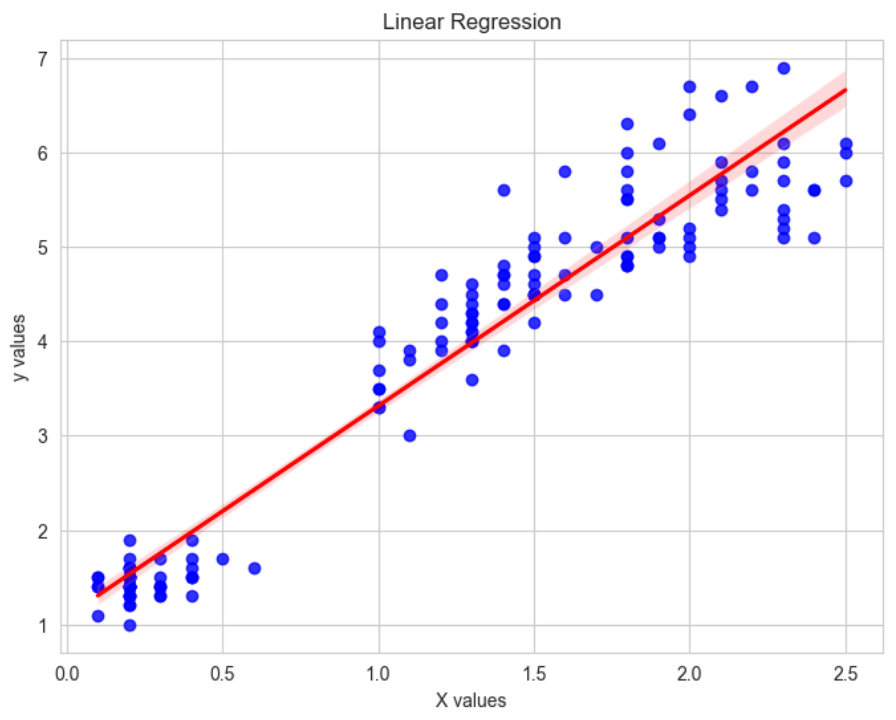
```
# VALIDATE MODEL USING LABELED TEST DATA

# Score model, used metric is model dependent, for regression: R^2
r2 = model.score(X_test, y_test)
print(f'R^2 : {r2:.3f}')

# Other metrics
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error,
root_mean_squared_error, r2_score

# Predict target feature for the labeled test data
y_test_pred = model.predict(X_test)

mae = mean_absolute_error(y_true=y_test, y_pred=y_test_pred)
mape = mean_absolute_percentage_error(y_true=y _test, y_pred=y_test_pred)
rmse = root_mean_squared_error(y _test, y_test_pred)
r2 = r2_score(y_test, y_test_pred)
print(f'MAE : {mae:.3f} - MAPE : {mape:.3f} - RMSE : {rmse:.3f} - R^2 : {r2:.3f}')
```

LINEAR REGRESSION WITH SCIKIT-LEARN

```
# APPLY MODEL ON NEW DATA
X_pred = …. (new feature data to predict the target feature for)
y_pred = model.predict(X_pred)
```

LINEAR REGRESSION WITH SCIKIT-LEARN

```
# SHOW REGRESSION LINE IN SCATTERPLOT
plt.figure(figsize=(8,6))
sns.regplot(x=X_train, y=y_train, scatter_kws={"color":"blue"}, line_kws={"color":"red"})
plt.title("Linear Regression")
plt.xlabel("X values")
plt.ylabel("y values")
plt.show()
```



LINEAR REGRESSION WITH SCIKIT-LEARN

Multiple predictors

```
# DATA PREPARATION
X = iris[['sepal_width', 'sepal_length', 'petal_width']] # Multiple predictors
y = iris['petal_length'] # Target feature to predict

# MODEL SELECTION AND HYPERPARAMETER SELECTION (MODEL SPECIFIC)
model = LinearRegression()

# SPLIT in TRAIN and TEST dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)

# DERIVE MODEL FROM LABELED DATA (TRAIN MODEL/FIT MODEL)
model.fit(X_train, y_train)

# DISPLAY COEFFICIENTS (intercept and slope)
print(f"Intercept: {model.intercept_}")
print(f"Coefficient: {model.coef_}")
```

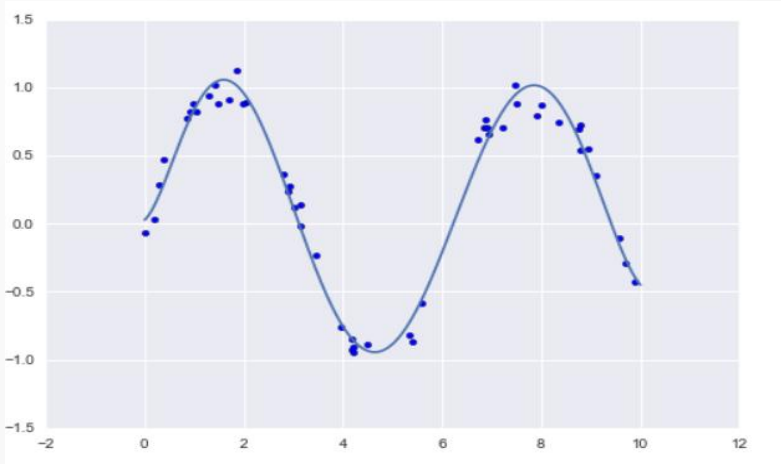
POLYNOMIAL REGRESSION

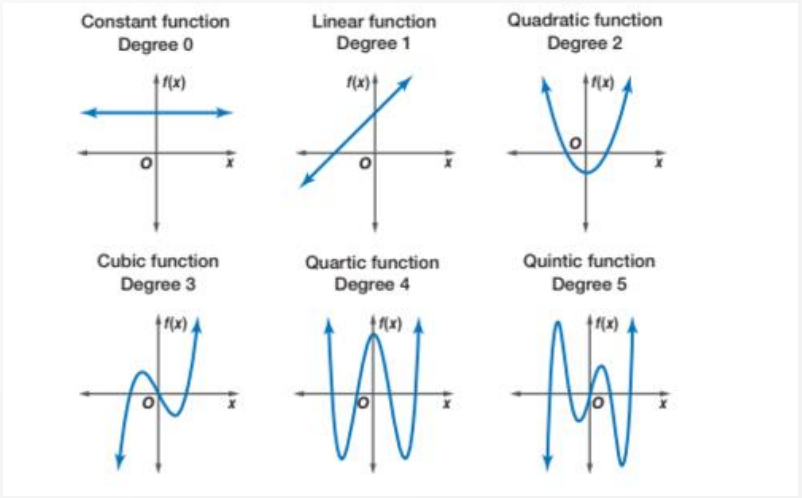
prediction of numerical features

Try to fit a polynomial function to the data in the best possible way

$y = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$

n: degree (order) of the polynomial function





Quadratic:

n = 2

$y = a_0 + a_1x + a_2x^2$

Cubic:

n = 3

$y = a_0 + a_1x + a_2x^2 + a_3x^3$

POLYNOMIAL REGRESSION WITH SCIKIT-LEARN

```
# MODEL SELECTION AND HYPERPARAMETER SELECTION (MODEL SPECIFIC)
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
model = make_pipeline(PolynomialFeatures(degree=2, include_bias=False), LinearRegression())

# DERIVE MODEL FROM LABELED DATA (TRAIN MODEL/FIT MODEL)
model.fit(X_train, y_train)

# VALIDATE MODEL USING LABELED TEST DATA
r2 = model.score(X_test, y_test)  # same for other scores

# APPLY MODEL ON NEW DATA
X_pred = …. (new feature data to predict the target feature for)
y_pred = model.predict(X_pred)
```