The UK's European university

University of Kent

COMP8270 / PROGRAMMING FOR ARTIFICIAL INTELLIGENCE

Fernando Otero
febo@kent.ac.uk
cs.kent.ac.uk/people/staff/febo

overview:

1. Regression

2. Python coding:
   - Linear Regression
   - Decision Tree Regressor

overview:

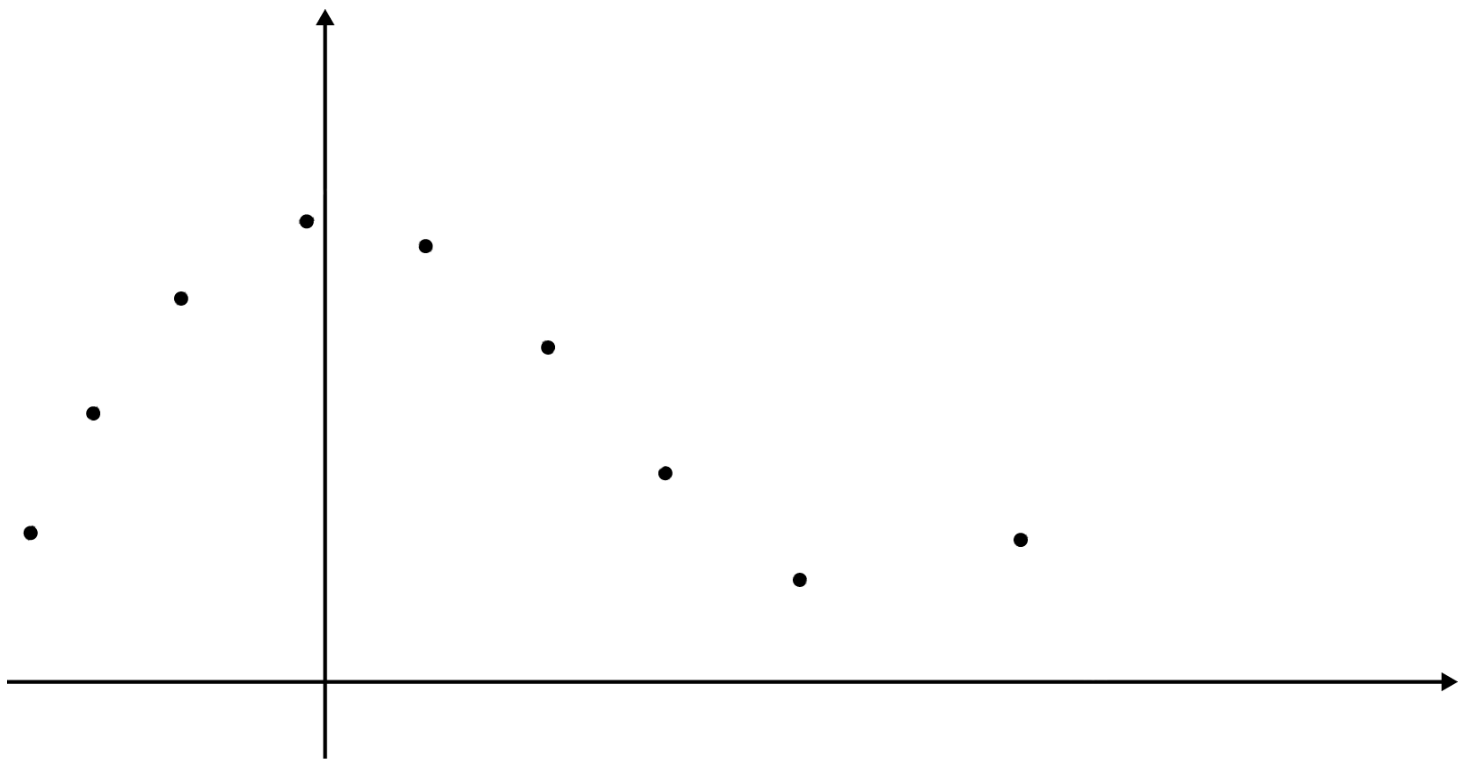
1. Regression

2. Python coding:
    - Linear Regression
    - Decision Tree Regressor

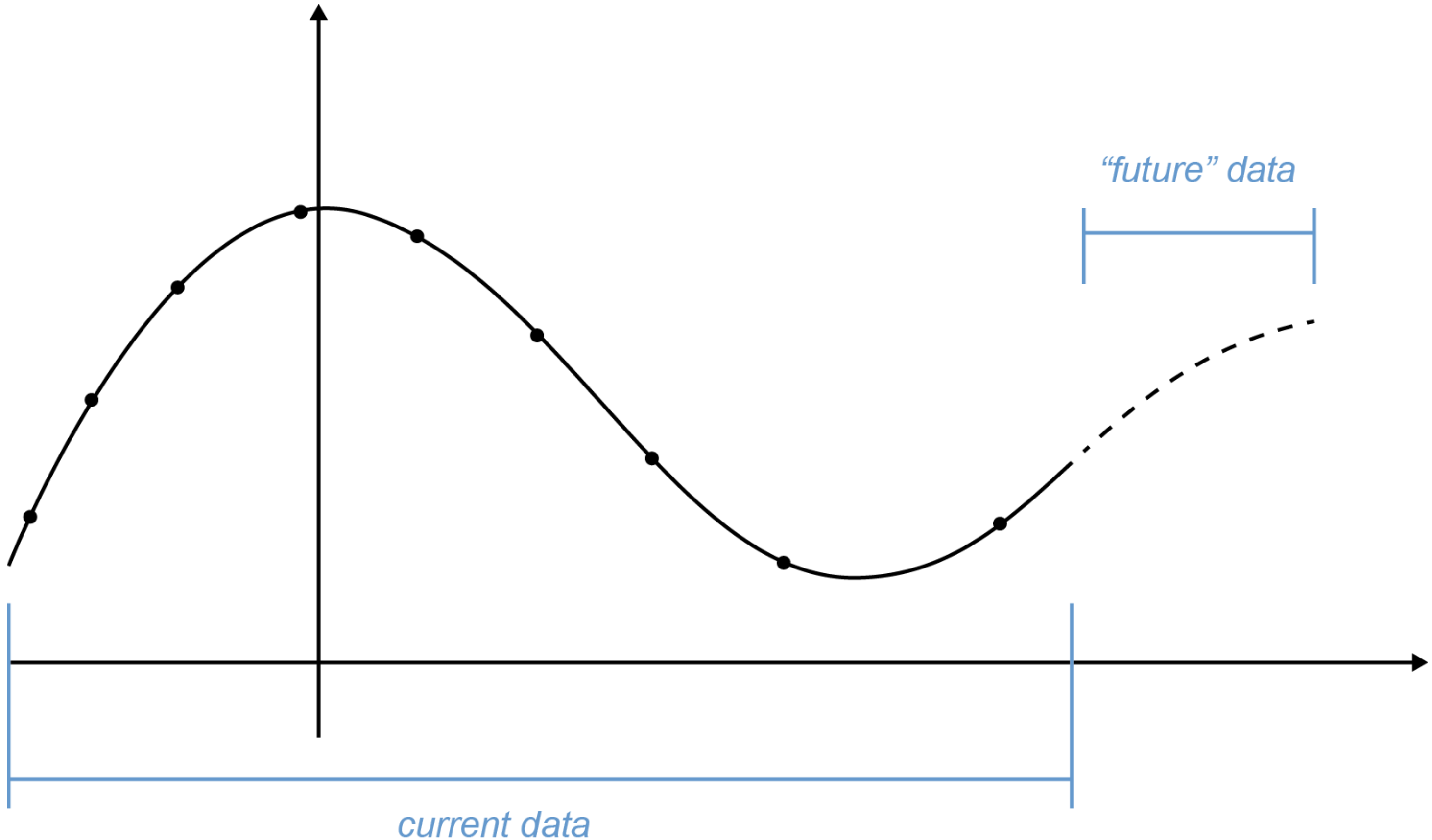
Regression task (1):

- Supervised learning (similar to classification)
- Consists of finding a model that maps a given input (attributes values) to a **numeric prediction**

Regression task (2):



Regression task (3):



Regression task (4):

Predictor Attributes				*Target Attribute*
fixed acidity	chlorides	citric acid	pH	quality
7.4	0.076	0.08	3.51	5
7.8	0.098	0.56	3.20	5
6.2	0.092	0.00	3.26	6
5.9	0.062	0.12	3.16	9
11.2	0.075	0.47	3.51	2
6.3	0.090	0.04	3.39	1

Regression task (5):

- Forecasting: predicting the economy growth based on market indicators

- Medical diagnosis: predicting the length of time a patient will live after undergoing a particular type of surgery

overview:

1. Regression

2. Python coding:
   - Linear Regression
   - Decision Tree Regressor

Regression with Python:

1. Choose your data

2. Pre-processing
   - Check missing values
   - Attribute transformation?

3. Create a regression model

4. Evaluate the performance

Example:

- Predicting concrete strength

```python
import pandas as pd


concrete_data = pd.read_csv(
    'https://cs.kent.ac.uk/~febo/COMP8270/Concrete_Data.csv'
)
```

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Concrete compressive strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30
...	...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28	44.28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28	31.18
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28	23.70
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28	32.77
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28	32.40

Pre-processing:

- Do we need any at this point?

```
# information regarding the attributes
concrete_data.info()

>  #   Column                         Non-Null Count  Dtype
> ---  ------                         --------------  -----
>  0   Cement                         1030 non-null   float64
>  1   Blast Furnace Slag             1030 non-null   float64
>  2   Fly Ash                        1030 non-null   float64
>  3   Water                          1030 non-null   float64
>  4   Superplasticizer               1030 non-null   float64
>  5   Coarse Aggregate               1030 non-null   float64
>  6   Fine Aggregate                 1030 non-null   float64
>  7   Age                            1030 non-null   int64
>  8   Concrete compressive strength  1030 non-null   float64
```

Pre-processing:

- Target attribute statistics:

```
# information regarding the target attribute
concrete_data.iloc[: , -1].describe()


> count    1030.000000
> mean       35.817961
> std        16.705742
> min         2.330000
> 25%        23.710000
> 50%        34.445000
> 75%        46.135000
> max        82.600000
> Name: Concrete compressive strength, dtype: float64
```

Model creation (1):

- Ordinary Least Squares

- Creates a model in the format:

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p$$

Model creation (2):

- Lineal Model:

```
from sklearn import linear_model

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)


# shows the coefficients of the linear model
regressor.coef_
```

Model creation (3):

- Decision Tree Regressor
  - Similar construction process as a classification Decision Tree
- Leaf nodes predict a continuous value

Model creation (4):

- Decision Tree:

```
from sklearn.tree import DecisionTreeRegressor


regressor =  DecisionTreeRegressor()
regressor.fit(X_train, y_train)
```

- You can visualise the tree the same way you visualise a classification Decision Tree

Evaluation (1):

- Residual (error) derived from the difference:

$$y_{true} - y_{predicted}$$

```
# score of the model (best possible score is 1.0)

regressor.score(X_train, y_train)
```

Evaluation (2):

- train + test partition:

```
from sklearn.model_selection import train_test_split

X = concrete_data.iloc[:, 0:-1]
y = concrete_data.iloc[:, -1]

# train + test split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

# builds the regressor
regressor =  DecisionTreeRegressor()
regressor.fit(X_train, y_train)

# evaluates on the test data
regressor.score(X_test, y_test)
```

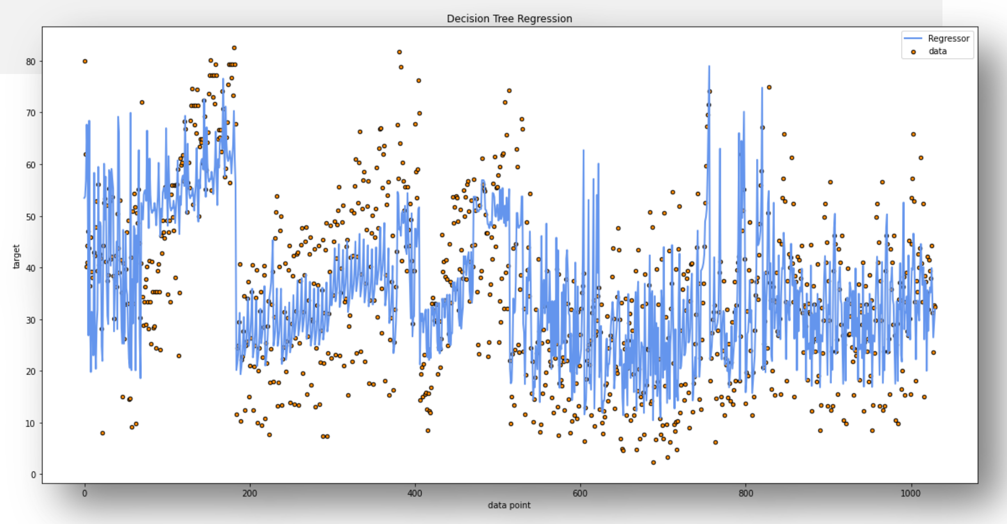
Visualisation:

```
import matplotlib.pyplot as plt

# predictions from the regressor
y = regressor.predict(X_train)
# data point indices
X = np.arange(len(y_train))

plt.figure(figsize=(20, 10))
plt.scatter(X, y_train, s=20, edgecolor="black", c="darkorange", label="data")
plt.plot(X, y, color="cornflowerblue", label="Regressor", linewidth=2)
plt.xlabel("data point")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()

plt.show()
```



Next lecture:

- Other AI techniques