Bevezetés a Deep Learningbe Python alapon

Michaletzky Tamás

Témavezető: Kurics Tamás

Mesterséges intelligencia

Gépi tanulás alatt azokat a számítógépes algoritmusokat értjük, melyek képesek tapasztalat útján tanulni.

Mély tanulás során neurális hálót használunk.

Mesterséges intelligencia

Gépi tanulás alatt azokat a számítógépes algoritmusokat értjük, melyek képesek tapasztalat útján tanulni.

Mély tanulás során neurális hálót használunk.

Modell épités: paraméterjavítás költségfüggvény alapján iteratívan → TANULÁS

Mesterséges intelligencia

Gépi tanulás alatt azokat a számítógépes algoritmusokat értjük, melyek képesek tapasztalat útján tanulni.

Mély tanulás során neurális hálót használunk.

Modell épités: paraméterjavítás költségfüggvény alapján iteratívan → TANULÁS

Cél: ismeretlen adathalmazon is jól működjön

Túltanulás

Paradigmaváltás

Hagyományos: Adat + Szabály → Válasz

MI: Adat + Válasz → Szabály

Paradigmaváltás

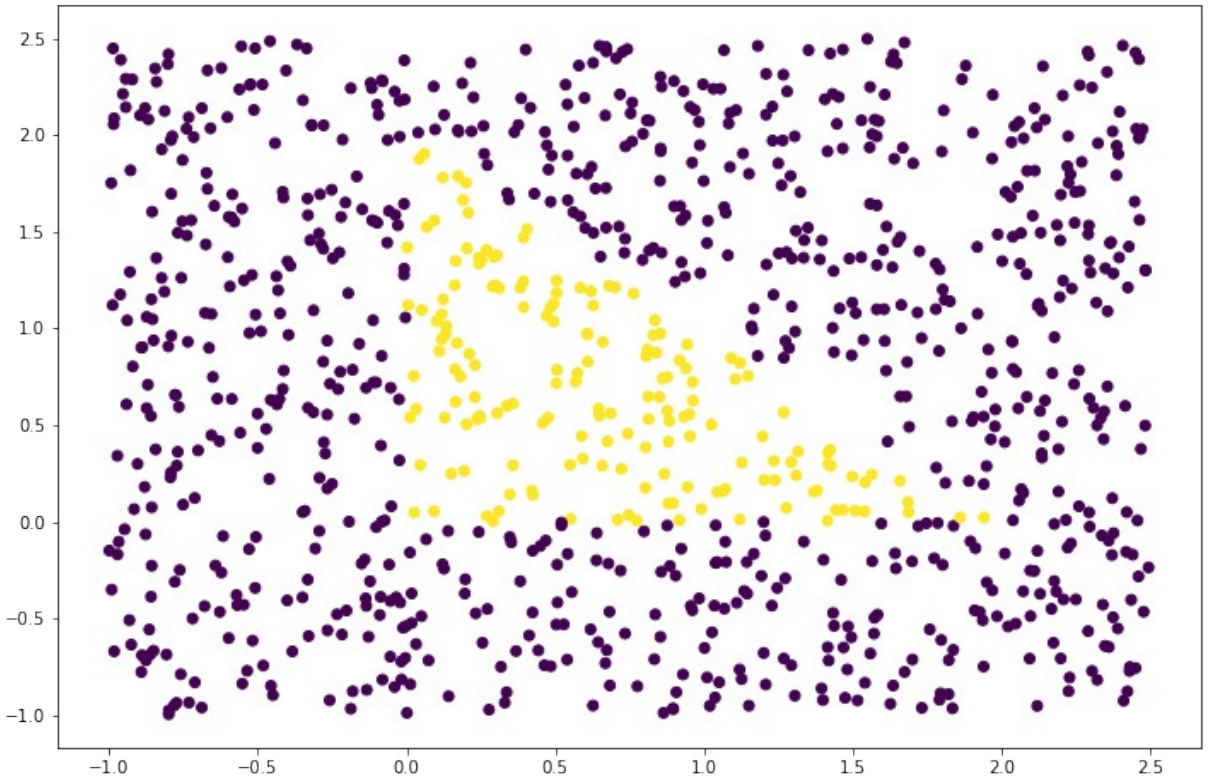
Hagyományos: Adat + Szabály → Válasz

MI: Adat + Válasz → Szabály

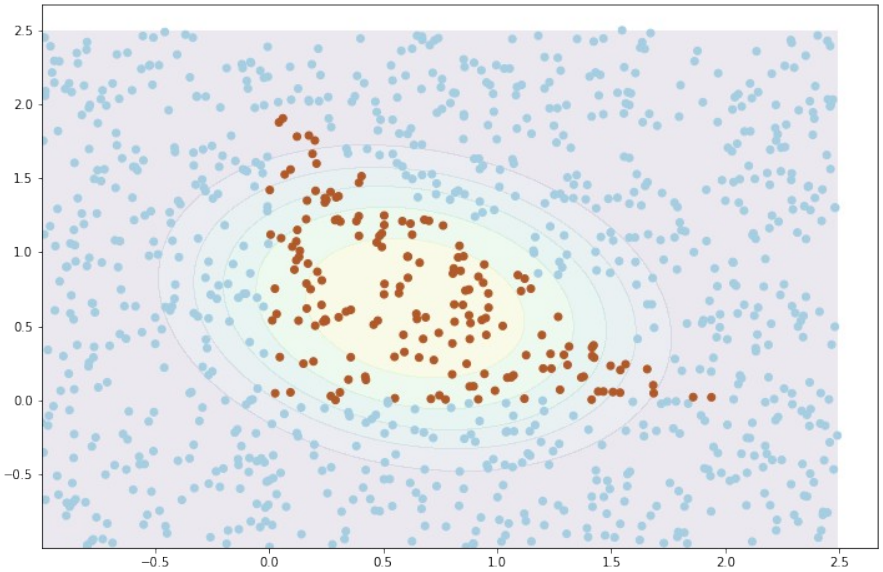
Matematika és programozás: Python

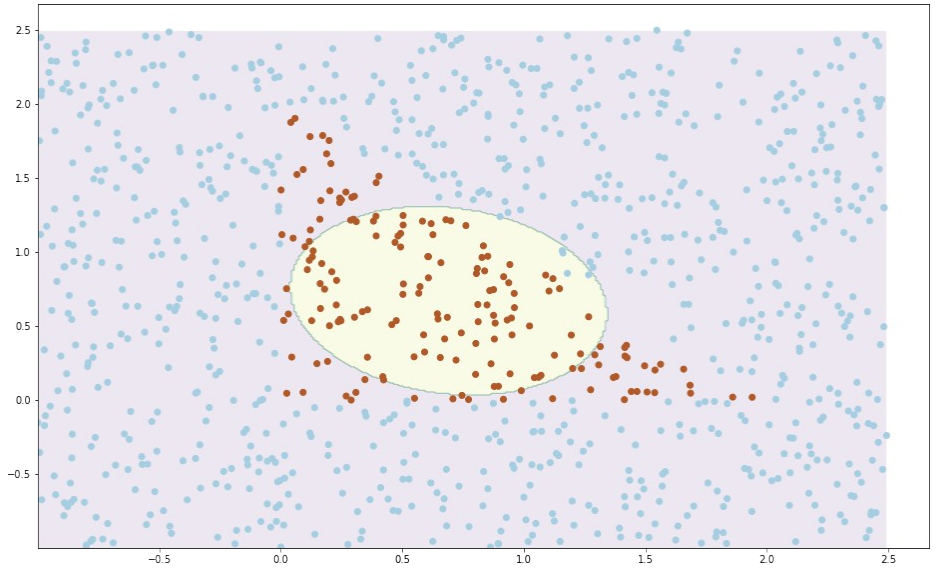
Tensorflow - Keras

Feladat:
Ismerjük fel a háromszöget!

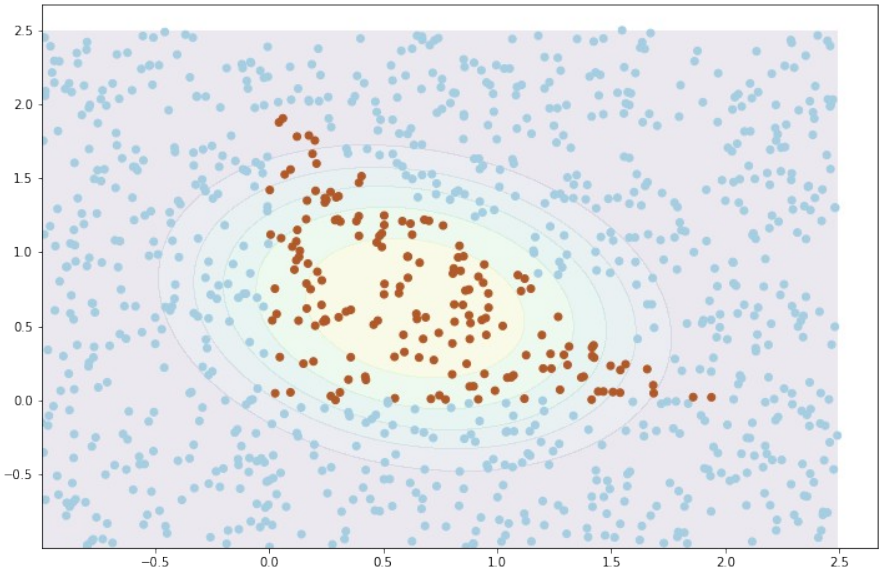


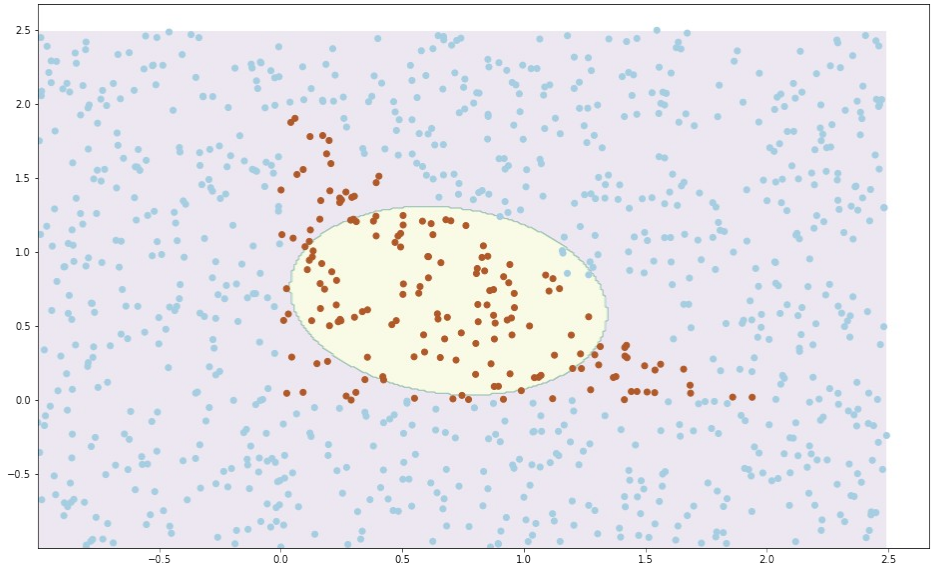
Logisztikus regresszióval





Logisztikus regresszióval

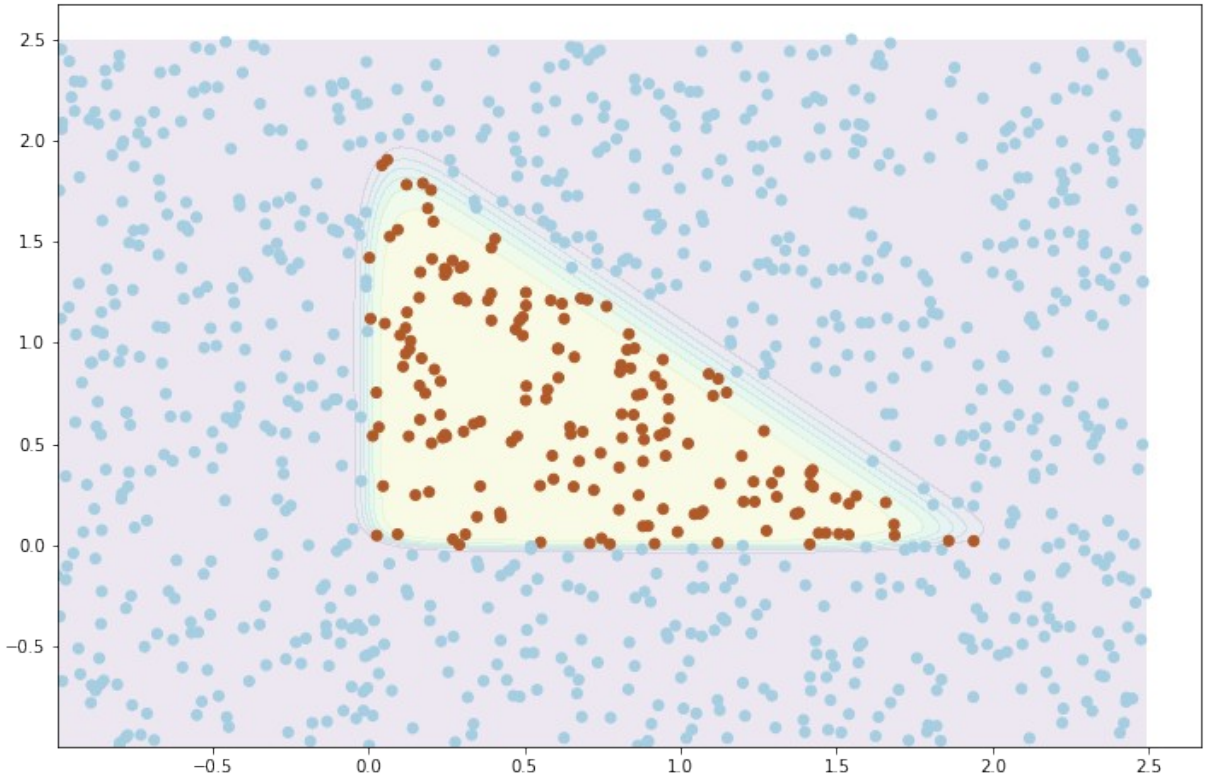




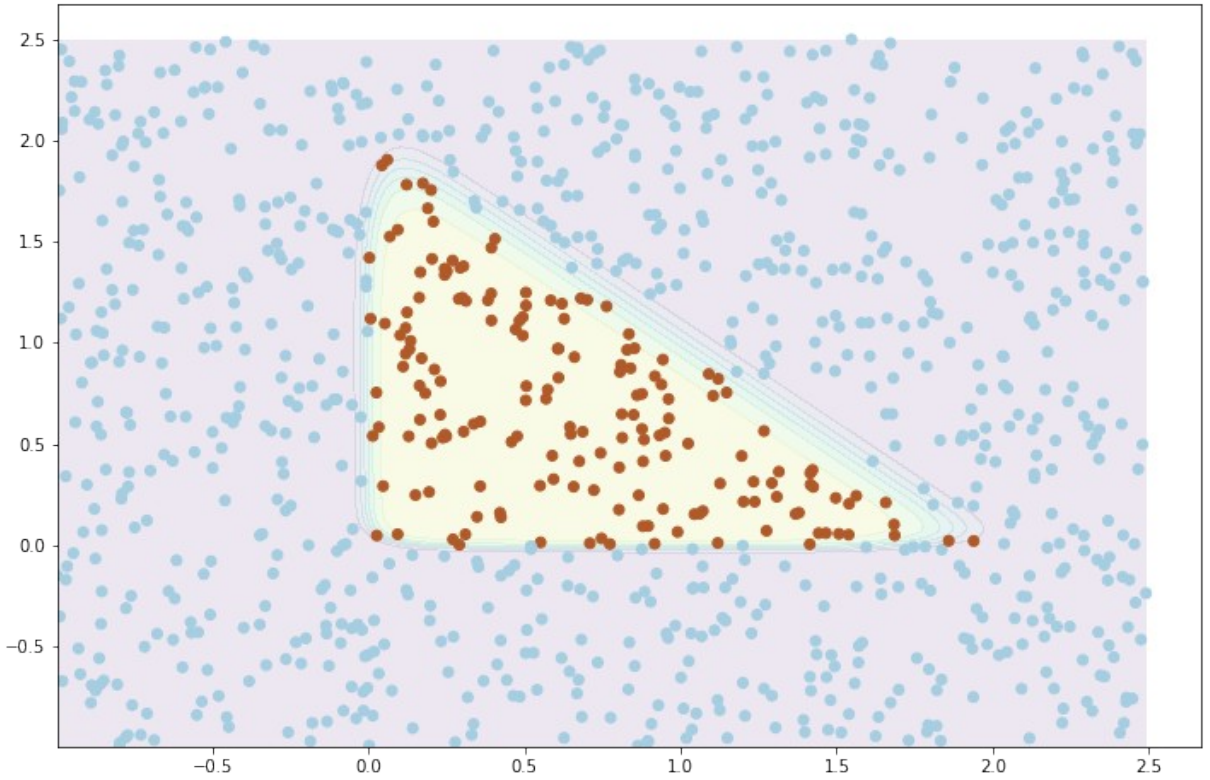
Pontosság: 0,927

Tévedés mátrix: 826, 4
69, 101

Mély tanulással



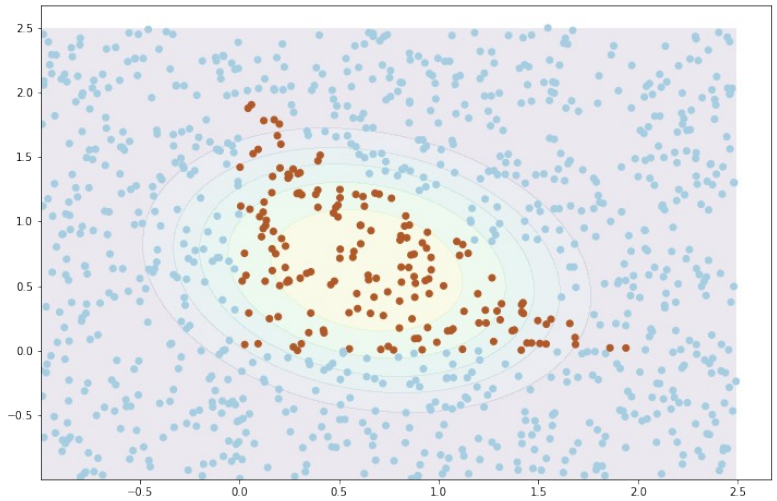
Mély tanulással



Pontosság: 0,99

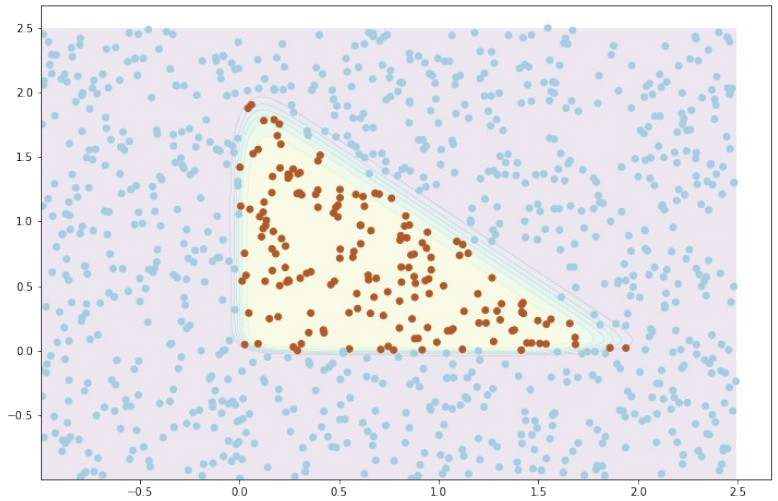
Tévedés mátrix: 824, 6
4, 166

Összehasonlítva



Pontosság: 0,927

Tévedés mátrix: 826, 4
69, 101



Pontosság: 0,99

Tévedés mátrix: 824, 6
4, 166

Köszönöm a figyelmet!

```
import numpy as np
import sklearn.metrics as mcs
import tensorflow.keras as keras

#Read data to variables X and y

#Logistic regression
model = keras.Sequential([
    keras.layers.Dense(units=1, activation='sigmoid', input_shape=(5,))
])
optimizer = keras.optimizers.SGD(learning_rate=0.01)

#Neural network
model = keras.Sequential([
    keras.layers.Input(shape=(2,)),
    keras.layers.Dense(units=3, activation='sigmoid'),
    keras.layers.Dense(units=1, activation='sigmoid')
])
optimizer = keras.optimizers.Adam(learning_rate=0.01)

#Same from here
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics='accuracy')
history = model.fit(X, y, verbose=0, epochs=200)

predictions = np.round(model.predict(X))
print(mcs.accuracy_score(predictions, y))
mcs.confusion_matrix(y, predictions)
```