**Harjoitukset**

Koneoppiminen, Syksy 2020

Tapani Alastalo

M1475

**Sisältö**

[1 Tehtävä 1 Lineaarinen regressio 3](#_Toc55996120)

[1.1 Lähdekoodit 4](#_Toc55996121)

[2 Tehtävä 2 5](#_Toc55996122)

[2.1 Lähdekoodit 6](#_Toc55996123)

[3 Tehtävä 3 8](#_Toc55996124)

[3.1 Lähdekoodi 8](#_Toc55996125)

[4 Tehtävät 4 Neuroverkot 10](#_Toc55996126)

[4.1 Lähdekoodi 10](#_Toc55996127)

[5 Tehtävä 5 12](#_Toc55996128)

[5.1 Lähdekoodi 12](#_Toc55996129)

[6 Tehtävä 6 14](#_Toc55996130)

[6.1 Lähdekoodi 15](#_Toc55996131)

[6.2 Lähdekoodi 17](#_Toc55996132)

[7 Tehtävä 7 19](#_Toc55996133)

[7.1 Lähdekoodi 19](#_Toc55996134)

[8 Tehtävä 8 21](#_Toc55996135)

[8.1 Lähdekoodi 21](#_Toc55996136)

[9 Tehtävä 9 23](#_Toc55996137)

[9.1 Lähdekoodi 24](#_Toc55996138)

[10 Tehtävä 10 26](#_Toc55996139)

[10.1 Lähdekoodi 27](#_Toc55996140)

[11 Tehtävä 11 28](#_Toc55996141)

[11.1 Lähdekoodi 30](#_Toc55996142)

[12 Tehtävä 12 31](#_Toc55996143)

[12.1 Lähdekoodi 33](#_Toc55996144)

[13 Tehtävä 13 34](#_Toc55996145)

[13.1 Lähdekoodi 34](#_Toc55996146)

[14 Tehtävä 14 34](#_Toc55996147)

[14.1 Lähdekoodi 34](#_Toc55996148)

[15 Tehtävä 15 36](#_Toc55996149)

[15.1 Lähdekoodi 37](#_Toc55996150)

[16 Tehtävä 16 38](#_Toc55996151)

[16.1 Lähdekoodi 39](#_Toc55996152)

[17 Tehtävä 17 40](#_Toc55996153)

[17.1 Lähdekoodi 40](#_Toc55996154)

[18 Tehtävä 18 43](#_Toc55996155)

[18.1 Lähdekoodi 43](#_Toc55996156)

[19 Tehtävä 19 46](#_Toc55996157)

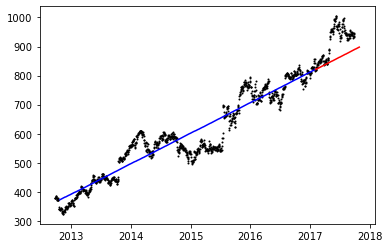
[19.1 Lähdekoodi 47](#_Toc55996158)

[20 Tehtävä 20: Lähtevien asiakkaiden tunnistaminen 49](#_Toc55996159)

[20.1 Lähdekoodi 49](#_Toc55996160)

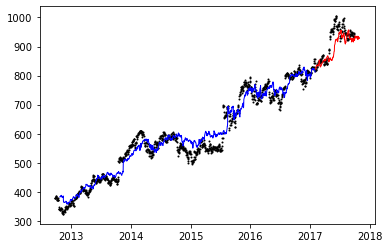
# Tehtävä 1 Lineaarinen regressio

a)



Ennusteen keskivirhe testidatassa on 70

b)



Ennusteen keskivirhe testidatassa on 35

## Lähdekoodit

import pandas as pd

import numpy as np

from sklearn import linear\_model

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

df = pd.read\_csv('data/Google\_Stock\_Price.csv')

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = df.apply(lambda row: len(df) - row.name, axis=1)

df['CloseFuture'] = df['Close'].shift(30)

df\_test = df[:185]

df\_train = df[185:]

X = np.array(df\_train[['Time', 'Close']])

y = np.array(df\_train['CloseFuture'])

model = linear\_model.LinearRegression()

model.fit(X, y)

ennuste\_train = model.predict(X)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Time', 'Close']])

ennuste\_test = model.predict(X\_test)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Date'].values, df['Close'].values, color='black', s=1)

plt.plot((df\_train['Date'] + pd.DateOffset(days=30)).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Date'] + pd.DateOffset(days=30)).values, df\_test['Ennuste'].values, color='red')

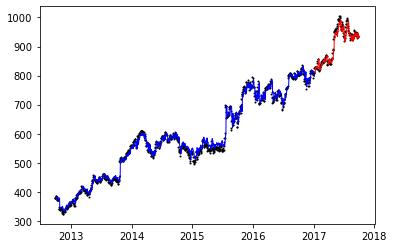
plt.show()

df\_validation = df\_test.dropna()

print("Ennusteen keskivirhe test datassa on %.f" % mean\_absolute\_error(df\_validation['CloseFuture'], df\_validation['Ennuste']))

# Tehtävä 2

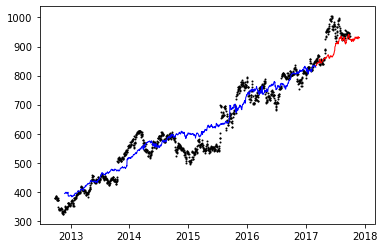
a) 7 päivän ennuste tulevaisuuteen



Ennusteen keskivirhe opetusdatassa on 16

Ennusteen keskivirhe testidatassa on 20

b) 60 päivän ennuste tulevaisuuteen



Ennusteen keskivirhe opetusdatassa on 34

Ennusteen keskivirhe testidatassa on 60

## Lähdekoodit

import pandas as pd

import numpy as np

from sklearn import linear\_model

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

days\_to\_forecast = 60

df = pd.read\_csv('data/Google\_Stock\_Price.csv')

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = df.apply(lambda row: len(df) - row.name, axis=1)

df['CloseFuture'] = df['Close'].shift(days\_to\_forecast)

df\_test = df[:185]

df\_train = df[185:]

X = np.array(df\_train[['Time', 'Close']])

y = np.array(df\_train['CloseFuture'])

model = linear\_model.LinearRegression()

model.fit(X, y)

ennuste\_train = model.predict(X)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Time', 'Close']])

ennuste\_test = model.predict(X\_test)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Date'].values, df['Close'].values, color='black', s=1)

plt.plot((df\_train['Date'] + pd.DateOffset(days=days\_to\_forecast)).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Date'] + pd.DateOffset(days=days\_to\_forecast)).values, df\_test['Ennuste'].values, color='red')

plt.show()

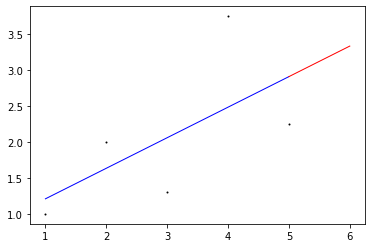
df\_train\_validation = df\_train.dropna()

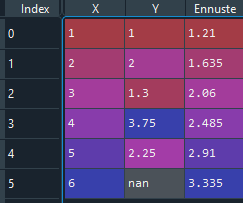
df\_test\_validation = df\_test.dropna()

print("Ennusteen keskivirhe opetus datassa on %.f" % mean\_absolute\_error(df\_train\_validation['CloseFuture'], df\_train\_validation['Ennuste']))

print("Ennusteen keskivirhe test datassa on %.f" % mean\_absolute\_error(df\_test\_validation['CloseFuture'], df\_test\_validation['Ennuste']))

# Tehtävä 3





## Lähdekoodi

import pandas as pd

import numpy as np

from sklearn import linear\_model

import matplotlib.pyplot as plt

data = [[1.00,1.00],[2.00,2.00],[3.00,1.30],[4.00,3.75],[5.00,2.25]] #,[6.00, None]]

df = pd.DataFrame(data, columns=['X', 'Y'])

df\_train = df[:]

df\_test = df[4:]

df\_test = df\_test.append({'X': 6.00}, ignore\_index=True)

#df\_train = df[:6]

#df\_test = df[4:]

X = np.array(df\_train['X'])

X = X.reshape(-1,1) # vain jos yksiulotteinen taulukko

y = np.array(df\_train['Y'])

model = linear\_model.LinearRegression()

model.fit(X, y)

df\_train['Ennuste'] = model.predict(X)

X\_test = np.array(df\_test['X'])

X\_test = X\_test.reshape(-1,1) # vain jos yksiulotteinen taulukko

df\_test['Ennuste'] = model.predict(X\_test)

plt.scatter(df['X'].values, df['Y'].values, color='black', s=1)

plt.plot(df\_train['X'].values, df\_train['Ennuste'].values, color='blue', linewidth=1)

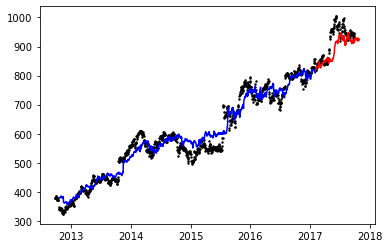
plt.plot(df\_test['X'].values, df\_test['Ennuste'].values, color='red', linewidth=1)

plt.show()

print('Mallin kertoimet ovat \n', model.coef\_, model.intercept\_)

df\_results = df\_train.append(df\_test.iloc[1:], ignore\_index = True)

# Tehtävät 4 Neuroverkot



Ennusteen keskivirhe opetusdatassa on 30

Ennusteen keskivirhe testidatassa on 40

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

import tensorflow as tf

from tensorflow import keras

from sklearn import preprocessing

days\_to\_forecast = 30

df = pd.read\_csv('data/Google\_Stock\_Price.csv')

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = df.apply(lambda row: len(df) - row.name, axis=1)

df['CloseFuture'] = df['Close'].shift(days\_to\_forecast)

df\_train = df[:185]

df\_test = df[185:]

X = np.array(df\_train[['Time', 'Close']])

scaler = preprocessing.MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

y = np.array(df\_train['CloseFuture'])

# luodaan sequential tyyppinen neuroverkkomalli

model = tf.keras.Sequential([

# määritellään neuroverkon piilotettu kerros. 10 neuronia (2 inputs), activation funktio = rectified lineaarifunction, input kerros (input\_shape) = input arvojen lukumäärä

keras.layers.Dense(10, activation='relu', input\_shape=(2,)),

# 2. piilotettu kerros

keras.layers.Dense(10, activation='relu'),

# mallin output kerros. 1 ulostulo (output). Ei aktivointifunktiota # , activation='softmax'

keras.layers.Dense(1)])

# optimointialgoritmi Adam algoritmi (learning rate=0.001).

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001), #'adam', #tf.train.AdamOptimizer(0.001),

loss='mse', #'categorical\_crossentropy',

metrics=['mae']) # ['accuracy'])

# epochs = kuinka monta kertaa opetusdata käydään läpi training vaiheessa (painotus), batch\_size = kuinka monen data rivin jälkeen painokertoimia päivitetään (oppiminen)

model.fit(X\_scaled, y, epochs = 100, batch\_size = 10)

ennuste\_train = model.predict(X\_scaled)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Time', 'Close']])

X\_testscaled = scaler.transform(X\_test)

ennuste\_test = model.predict(X\_testscaled)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Date'].values, df['Close'].values, color='black', s=2)

plt.plot((df\_train['Date'] + pd.DateOffset(days=30)).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Date'] + pd.DateOffset(days=30)).values, df\_test['Ennuste'].values, color='red')

plt.show()

df\_train\_validation = df\_train.dropna()

df\_test\_validation = df\_test.dropna()

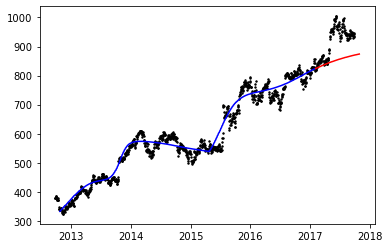
print("Ennusteen keskivirhe opetusdatassa on %.f" %

mean\_absolute\_error(df\_train\_validation['CloseFuture'], df\_train\_validation['Ennuste']))

print("Ennusteen keskivirhe testidatassa on %.f" %

mean\_absolute\_error(df\_test\_validation['CloseFuture'], df\_test\_validation['Ennuste']))

# Tehtävä 5



Ennusteen keskivirhe opetusdatassa on 19

Ennusteen keskivirhe testidatassa on 74

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

import tensorflow as tf

from tensorflow import keras

from sklearn import preprocessing

days\_to\_forecast = 30

df = pd.read\_csv('data/Google\_Stock\_Price.csv')

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = df.apply(lambda row: len(df) - row.name, axis=1)

df['CloseFuture'] = df['Close'].shift(days\_to\_forecast)

df\_train = df[:185]

df\_test = df[185:]

X = np.array(df\_train[['Time']])

X = X.reshape(-1,1)

scaler = preprocessing.MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

y = np.array(df\_train['CloseFuture'])

# luodaan sequential tyyppinen neuroverkkomalli

model = tf.keras.Sequential([

# määritellään neuroverkon piilotettu kerros. 10 neuronia (1 input), activation funktio = sigmoid, input kerros (input\_shape) = input arvojen lukumäärä

keras.layers.Dense(20, activation='sigmoid', input\_shape=(1,)),

# 2. piilotettu kerros

keras.layers.Dense(20, activation='sigmoid'),

# 3. piilotettu kerros

keras.layers.Dense(20, activation='relu'),

# mallin output kerros. 1 ulostulo (output). Ei aktivointifunktiota # , activation='softmax'

keras.layers.Dense(1)])

# optimointialgoritmi Adam algoritmi (learning rate=0.001).

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.01), #'adam', #tf.train.AdamOptimizer(0.001),

loss='mse', #'categorical\_crossentropy',

metrics=['mae']) # ['accuracy'])

# epochs = kuinka monta kertaa opetusdata käydään läpi training vaiheessa (painotus), batch\_size = kuinka monen data rivin jälkeen painokertoimia päivitetään (oppiminen)

model.fit(X\_scaled, y, epochs = 100, batch\_size = 10)

ennuste\_train = model.predict(X\_scaled)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Time']])

X\_test = X\_test.reshape(-1,1)

X\_testscaled = scaler.transform(X\_test)

ennuste\_test = model.predict(X\_testscaled)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Date'].values, df['Close'].values, color='black', s=2)

plt.plot((df\_train['Date'] + pd.DateOffset(days=30)).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Date'] + pd.DateOffset(days=30)).values, df\_test['Ennuste'].values, color='red')

plt.show()

df\_train\_validation = df\_train.dropna()

df\_test\_validation = df\_test.dropna()

print("Ennusteen keskivirhe opetusdatassa on %.f" %

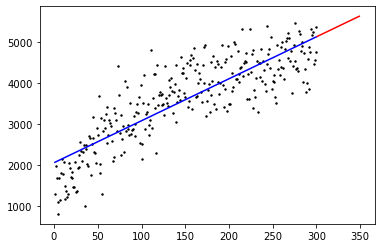
mean\_absolute\_error(df\_train\_validation['CloseFuture'], df\_train\_validation['Ennuste']))

print("Ennusteen keskivirhe testidatassa on %.f" %

mean\_absolute\_error(df\_test\_validation['CloseFuture'], df\_test\_validation['Ennuste']))ss

# Tehtävä 6

a) Lineaarista regressiomallia, jonka input-muuttujana on pelkkä aika.



Ennusteen keskivirhe opetusdatassa on 452

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

from sklearn import linear\_model

df = pd.read\_csv('data/Kysynta.csv', sep=';', encoding='latin\_1')

print(df)

df\_train = df[:300]

df\_test = df[300:]

for i in range(301, 350):

df\_test = df\_test.append({'Päivä': i}, ignore\_index=True)

X = np.array(df\_train[['Päivä']])

X = X.reshape(-1,1)

y = np.array(df\_train['Kysyntä'])

model = linear\_model.LinearRegression()

model.fit(X, y)

ennuste\_train = model.predict(X)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Päivä']])

X\_test = X\_test.reshape(-1,1)

ennuste\_test = model.predict(X\_test)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Päivä'].values, df['Kysyntä'].values, color='black', s=2)

plt.plot((df\_train['Päivä']).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Päivä']).values, df\_test['Ennuste'].values, color='red')

plt.show()

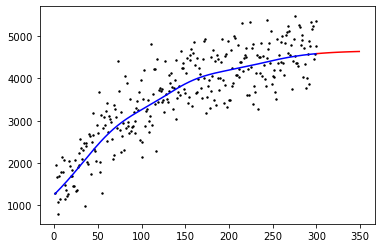
df\_train\_validation = df\_train.dropna()

df\_test\_validation = df\_test.dropna()

print("Ennusteen keskivirhe opetusdatassa on %.f" %

mean\_absolute\_error(df\_train\_validation['Kysyntä'], df\_train\_validation['Ennuste']))

b) MLP-neuroverkkoa, jonka input-muuttujana on pelkkä aika.



Ennusteen keskivirhe opetusdatassa on 405

Ennusteen keskivirhe testidatassa on 407

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_absolute\_error

import tensorflow as tf

from tensorflow import keras

from sklearn import preprocessing

df = pd.read\_csv('data/Kysynta.csv', sep=';', encoding='latin\_1')

print(df)

df\_train = df[:300]

df\_test = df[300:]

for i in range(301, 350):

df\_test = df\_test.append({'Päivä': i}, ignore\_index=True)

X = np.array(df\_train[['Päivä']])

X = X.reshape(-1,1)

scaler = preprocessing.MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

y = np.array(df\_train['Kysyntä'])

# luodaan sequential tyyppinen neuroverkkomalli

model = tf.keras.Sequential([

# määritellään neuroverkon piilotettu kerros. 10 neuronia (1 input), activation funktio = sigmoid, input kerros (input\_shape) = input arvojen lukumäärä

keras.layers.Dense(20, activation='sigmoid', input\_shape=(1,)),

# 2. piilotettu kerros

keras.layers.Dense(20, activation='tanh'),

# 3. piilotettu kerros

keras.layers.Dense(20, activation='relu'),

# mallin output kerros. 1 ulostulo (output). Ei aktivointifunktiota # , activation='softmax'

keras.layers.Dense(1)])

# optimointialgoritmi Adam algoritmi (learning rate=0.001).

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.01),

loss='mse',

metrics=['mae'])

# epochs = kuinka monta kertaa opetusdata käydään läpi training vaiheessa (painotus), batch\_size = kuinka monen data rivin jälkeen painokertoimia päivitetään (oppiminen)

model.fit(X\_scaled, y, epochs = 100, batch\_size = 10)

ennuste\_train = model.predict(X\_scaled)

df\_train['Ennuste'] = ennuste\_train

X\_test = np.array(df\_test[['Päivä']])

X\_test = X\_test.reshape(-1,1)

X\_testscaled = scaler.transform(X\_test)

ennuste\_test = model.predict(X\_testscaled)

df\_test['Ennuste'] = ennuste\_test

plt.scatter(df['Päivä'].values, df['Kysyntä'].values, color='black', s=2)

plt.plot((df\_train['Päivä']).values, df\_train['Ennuste'].values, color='blue')

plt.plot((df\_test['Päivä']).values, df\_test['Ennuste'].values, color='red')

plt.show()

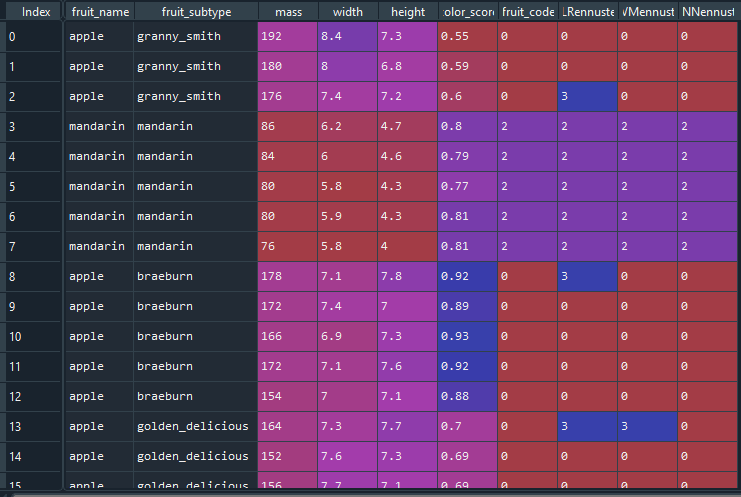
df\_train\_validation = df\_train.dropna()

df\_test\_validation = df\_test.dropna()

print("Ennusteen keskivirhe opetusdatassa on %.f" %

mean\_absolute\_error(df\_train\_validation['Kysyntä'], df\_train\_validation['Ennuste']))

# Tehtävä 7



Ennusteen keskivirhe logistisella regressiolla on 0.881.

Ennusteen keskivirhe SVM luokittelulla on 0.966.

Ennusteen keskivirhe K-Lähimmännaapurin luokittelulla on 0.983.

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import linear\_model

from sklearn.metrics import accuracy\_score

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

df = pd.read\_csv('data/fruit\_data.csv', sep=',', encoding='utf-8')

X = np.array(df[['mass', 'width', 'height', 'color\_score']])

fruit\_codes = {'apple':0, 'lemon':1, 'mandarin':2, 'orange':3}

df['fruit\_code'] = df['fruit\_name'].map(fruit\_codes)

y = np.array(df['fruit\_code'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# LOGISTIC REGRESSION

model = linear\_model.LogisticRegression(multi\_class='multinomial', solver='newton-cg')

model.fit(X\_scaled, y)

ennuste = model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df['LRennuste'] = ennuste

# SUPPORT VECTOR CLASSIFIER

model = SVC()

model.fit(X\_scaled, y)

ennuste = model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df['SVMennuste'] = ennuste

# SUPPORT VECTOR CLASSIFIER

model = KNeighborsClassifier()

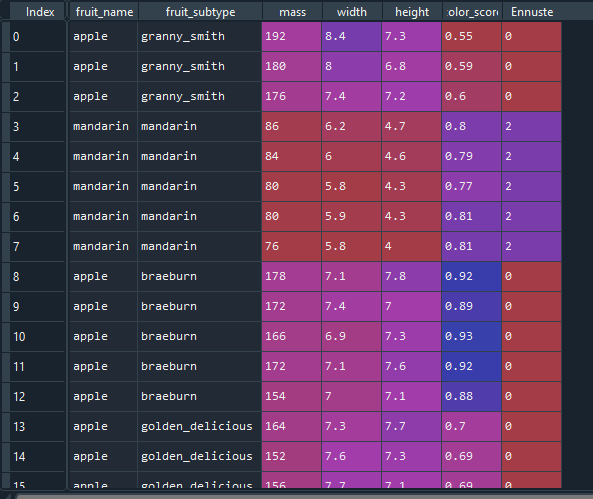
model.fit(X\_scaled, y)

ennuste = model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df['KNNennuste'] = ennuste

# Tehtävä 8



Osumatarkkuus = 100 % harjoitusdatalla, kun käytetään 20 opetuskertaa.

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

import tensorflow as tf

from tensorflow import keras

df = pd.read\_csv('data/fruit\_data.csv', sep=',', encoding='utf-8')

X = np.array(df[['mass', 'width', 'height', 'color\_score']])

y = np.array(pd.get\_dummies(df['fruit\_name']))

# Skaalataan X arvot keskiarvoon 0 ja keskihajontaan 1

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = keras.Sequential([

# 1. piilotettu / input kerros

keras.layers.Dense(30, activation=tf.nn.relu, input\_shape=(X\_scaled.shape[1],)),

# 2. piilotetu kerros

keras.layers.Dense(30, activation=tf.nn.relu),

# output kerros -> 4 output luokkaa, softmax tulostaa ko. luokan todennäköisyyden

keras.layers.Dense(4, activation=tf.nn.softmax)

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['categorical\_accuracy'])

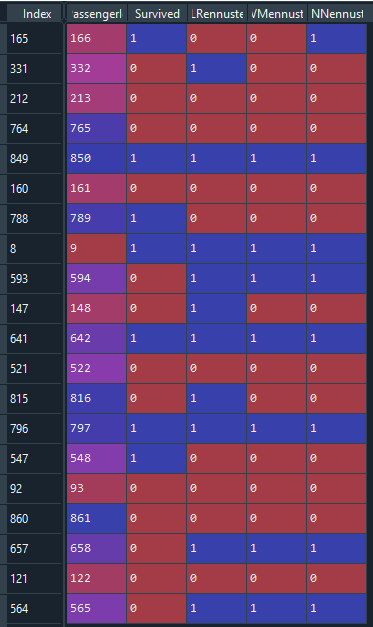
model.fit(X\_scaled, y, epochs=20, batch\_size=1)

# hakee sarakkeesta ennusteen, jonka todennäköisyys suurin

ennuste = np.argmax(model.predict(X\_scaled), axis=1)

df['Ennuste'] = ennustess

# Tehtävä 9



Ennusteen keskivirhe logistisella regressiolla on 0.781.

Ennusteen keskivirhe SVM luokittelulla on 0.800.

Ennusteen keskivirhe K-Lähimmännaapurin luokittelulla on 0.781.

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn import linear\_model

from sklearn.metrics import accuracy\_score

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

df = pd.read\_csv('data/Titanic.csv', sep=',', encoding='utf-8')

sex\_B = {'male':0, 'female':1}

df['Sex\_B'] = df['Sex'].map(sex\_B)

df['Age'].fillna(-1, inplace=True)

embarked\_B = {'C':0, 'S':1, 'Q':2}

df['Embarked\_B'] = df['Embarked'].map(embarked\_B)

df['Embarked\_B'].fillna(-1, inplace=True)

#print(df['Pclass'].unique())

#for col in df:

# print(col)

# print(df[col].unique())

df\_train = df.sample(n = 200, replace = False)

df\_test = df.drop(df\_train.index)

input\_variables = ['Pclass', 'Sex\_B', 'Age', 'SibSp', 'Parch', 'Embarked\_B']

# LOGISTIC REGRESSION

# train

X = np.array(df\_train[input\_variables])

y = np.array(df\_train['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = linear\_model.LogisticRegression(multi\_class='multinomial', solver='newton-cg')

model.fit(X\_scaled, y)

# test

X = np.array(df\_test[input\_variables])

y = np.array(df\_test['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

ennuste = model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df\_test['LRennuste'] = ennuste

# SUPPORT VECTOR CLASSIFIER

# train

X = np.array(df\_train[input\_variables])

y = np.array(df\_train['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = SVC()

model.fit(X\_scaled, y)

# test

X = np.array(df\_test[input\_variables])

y = np.array(df\_test['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

ennuste = model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df\_test['SVMennuste'] = ennuste

# SUPPORT VECTOR CLASSIFIER

# train

X = np.array(df\_train[input\_variables])

y = np.array(df\_train['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = KNeighborsClassifier()

model.fit(X\_scaled, y)

# test

X = np.array(df\_test[input\_variables])

y = np.array(df\_test['Survived'])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

ennuste = model.predict(X\_scaled)

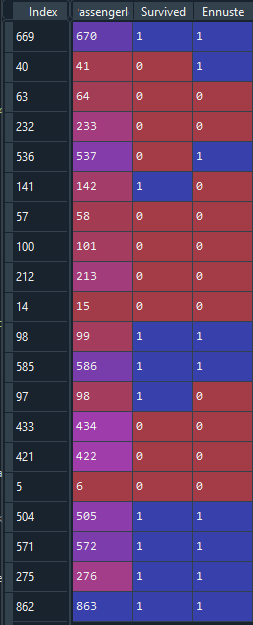
print(accuracy\_score(y, ennuste))

df\_test['KNNennuste'] = ennuste

results\_fields = ['PassengerId', 'Survived', 'LRennuste', 'SVMennuste', 'KNNennuste']

df\_results = df\_test[results\_fields].sample(20)

# Tehtävä 10



Ennusteen tarkkuus on 0.855.

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

import tensorflow as tf

from tensorflow import keras

df = pd.read\_csv('data/Titanic.csv', sep=',', encoding='utf-8')

sex\_B = {'male':0, 'female':1}

df['Sex\_B'] = df['Sex'].map(sex\_B)

df['Age'].fillna(-1, inplace=True)

embarked\_B = {'C':0, 'S':1, 'Q':2}

df['Embarked\_B'] = df['Embarked'].map(embarked\_B)

df['Embarked\_B'].fillna(-1, inplace=True)

df\_train = df.sample(n = 200, replace = False)

df\_test = df.drop(df\_train.index)

input\_variables = ['Pclass', 'Sex\_B', 'Age', 'SibSp', 'Parch', 'Embarked\_B']

# train

X = np.array(df\_train[input\_variables])

y = np.array(pd.get\_dummies(df\_train['Survived']))

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = keras.Sequential([

# 1. piilotettu / input kerros

keras.layers.Dense(30, activation=tf.nn.relu, input\_shape=(X\_scaled.shape[1],)),

# 2. piilotetu kerros

keras.layers.Dense(30, activation=tf.nn.relu),

# output kerros -> 4 output luokkaa, softmax tulostaa ko. luokan todennäköisyyden

keras.layers.Dense(2, activation=tf.nn.softmax)

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['categorical\_accuracy'])

model.fit(X\_scaled, y, epochs=20, batch\_size=1)

# test

X = np.array(df\_test[input\_variables])

#y = np.array(pd.get\_dummies(df\_test['Survived']))

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# hakee sarakkeesta ennusteen, jonka todennäköisyys suurin

ennuste = np.argmax(model.predict(X\_scaled), axis=1)

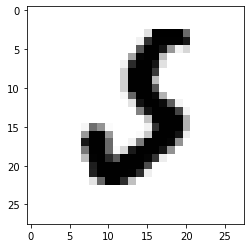
df\_test['Ennuste'] = ennuste

results\_fields = ['PassengerId', 'Survived', 'Ennuste']

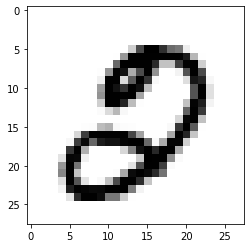
df\_results = df\_test[results\_fields].sample(20)ss

# Tehtävä 11

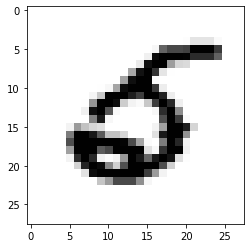
Kategorointitarkkuus opetusdatassa saavutti 99.5% tarkkuuden ja testidatassa 98.0 % tarkkuuden.



Malli tunnistaa yllä olevan kuvan 37 % todennäköisyydellä numeroksi 3. 31 % todennäköisyydellä numeroksi 5 ja 30 % todennäköisyydellä numeroksi 6. Ihmissilmä näkee tuossa numeron 5.



Malli tunnistaa yllä olevan kuvan 64 % todennäköisyydellä numeroksi 0 ja 36 % todennäköisyydellä numeroksi 2. Ihmissilmä näkee tuossa numeron 2 hahmotelmaa.



Malli tunnistaa yllä olevan kuvan 50 % todennäköisyydellä numeroksi 3, 30 % todennäköisyydellä numeroksi 6 ja 19 % todennäköisyydellä numeroksi 5. Ihmissilmä näkee tuossa numeroa 5 ja 6.

## Lähdekoodi

import tensorflow as tf

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

# spyderin blokki ajo

#%%

plt.imshow(x\_train[0], cmap='Greys')

#%%

# tehdään x\_train taulukosta 1-ulotteinen

x\_train\_flat = x\_train.reshape(60000, 784)

x\_test\_flat = x\_test.reshape(10000, 784)

# skaalaus

x\_train\_flat = x\_train\_flat/255

x\_test\_flat = x\_test\_flat/255

y\_train = pd.get\_dummies(y\_train)

y\_test = pd.get\_dummies(y\_test)

#%%

model = tf.keras.Sequential([

tf.keras.layers.Dense(1000, activation='relu', input\_shape=(x\_train\_flat.shape[1],)),

tf.keras.layers.Dense(100, activation='relu'),

# softmax muuttaa lopputuloksen luokkien todennäköisyydeksi

tf.keras.layers.Dense(10, activation='softmax')

])

model.compile(loss='categorical\_crossentropy',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

metrics=['categorical\_accuracy'])

model.fit(x\_train\_flat, y\_train, validation\_data=(x\_test\_flat, y\_test), epochs=10, batch\_size=100)

#%%

ennuste\_test = model.predict(x\_test\_flat)

#%%

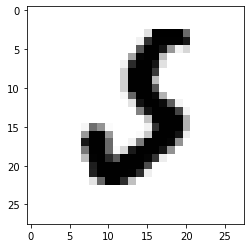
plt.imshow(x\_test[9749], cmap='Greys')

plt.imshow(x\_test[9768], cmap='Greys')

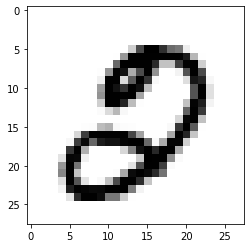
plt.imshow(x\_test[9982], cmap='Greys')

# Tehtävä 12

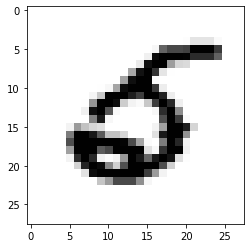
Kategorointitarkkuus opetusdatassa saavutti 99.6% tarkkuuden ja testidatassa 99.1 % tarkkuuden.



Käytetty malli tunnistaa yllä olevan kuvan 98 % todennäköisyydellä numeroksi 5. Edellinen malli tunnistaa yllä olevan kuvan 37 % todennäköisyydellä numeroksi 3, 31 % todennäköisyydellä numeroksi 5 ja 30 % todennäköisyydellä numeroksi 6. Ihmissilmä näkee tuossa numeron 5.



Malli tunnistaa yllä olevan kuvan 99 % todennäköisyydellä numeroksi 2, kun se edellisellä mallilla tunnistettiin 64 % todennäköisyydellä numeroksi 0.



Malli tunnistaa yllä olevan kuvan 99 % todennäköisyydellä numeroksi 5, kun edellinen malli tunnistaa yllä olevan kuvan 50 % todennäköisyydellä numeroksi 3.

## Lähdekoodi

import tensorflow as tf

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

plt.imshow(x\_train[0], cmap='Greys')

# tehdään x\_train taulukosta 1-ulotteinen, säilytetään koko ja määritellään mustavalkoiseksi (1)

x\_train\_flat = x\_train.reshape(60000, 28, 28, 1)

x\_test\_flat = x\_test.reshape(10000, 28, 28, 1)

x\_train\_flat = x\_train\_flat/255

x\_test\_flat = x\_test\_flat/255

y\_train = pd.get\_dummies(y\_train)

y\_test = pd.get\_dummies(y\_test)

model = tf.keras.Sequential([

tf.keras.layers.Conv2D(30, kernel\_size=5, activation='relu', input\_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D(pool\_size=2, strides=2),

tf.keras.layers.Conv2D(15, kernel\_size=5, activation='relu'),

tf.keras.layers.MaxPooling2D(pool\_size=2, strides=2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(1000, activation='relu'),

tf.keras.layers.Dense(50, activation='relu'),

tf.keras.layers.Dense(10, activation='softmax')

])

model.compile(loss='categorical\_crossentropy',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

metrics=['categorical\_accuracy'])

model.fit(x\_train\_flat, y\_train, validation\_data=(x\_test\_flat, y\_test), epochs=10, batch\_size=100)

ennuste\_test = model.predict(x\_test\_flat)

plt.imshow(x\_test[268], cmap='Greys')

plt.imshow(x\_test[9749], cmap='Greys')

plt.imshow(x\_test[9768], cmap='Greys')

plt.imshow(x\_test[9982], cmap='Greys')

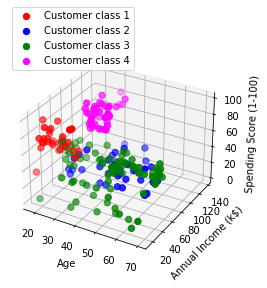
# Tehtävä 13

Sd

## Lähdekoodi

sds

# Tehtävä 14



## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from mpl\_toolkits.mplot3d import Axes3D

df = pd.read\_csv('data/Mall\_Customers.csv')

fields = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

X = np.array(df[fields])

model = KMeans(n\_clusters=4)

model.fit(X)

labels = model.labels\_

df['Label'] = labels

colors = {0:'red', 1:'blue', 2:'green', 3:'magenta'}

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

for i in range(0,4):

x = df.loc[df['Label'] == i][fields[0]].values

y = df.loc[df['Label'] == i][fields[1]].values

z = df.loc[df['Label'] == i][fields[2]].values

ax.scatter(x, y, z, marker='o', s=40, color=colors[i], label='Customer class '+str(i+1))

ax.set\_xlabel(fields[0])

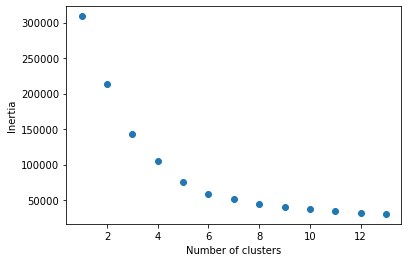
ax.set\_ylabel(fields[1])

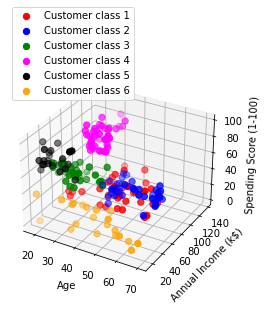
ax.set\_zlabel(fields[2])

ax.legend(loc='upper left', bbox\_to\_anchor=(0.0, 1.2))

plt.show()

# Tehtävä 15





## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from mpl\_toolkits.mplot3d import Axes3D

df = pd.read\_csv('data/Mall\_Customers.csv')

fields = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']

X = np.array(df[fields])

inertia = []

for i in range(1,14):

model = KMeans(n\_clusters=i)

model.fit(X)

inertia.append(model.inertia\_)

plt.scatter(np.arange(1,14), inertia)

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()

model = KMeans(n\_clusters=6)

model.fit(X)

labels = model.labels\_

df['Label'] = labels

colors = {0:'red', 1:'blue', 2:'green', 3:'magenta', 4:'black', 5:'orange'}

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

for i in range(0,6):

x = df.loc[df['Label'] == i][fields[0]].values

y = df.loc[df['Label'] == i][fields[1]].values

z = df.loc[df['Label'] == i][fields[2]].values

ax.scatter(x, y, z, marker='o', s=40, color=colors[i], label='Customer class '+str(i+1))

ax.set\_xlabel(fields[0])

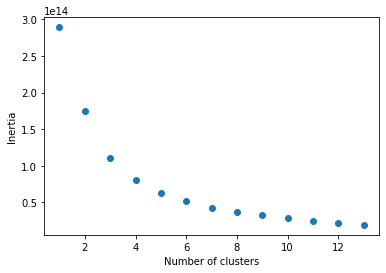
ax.set\_ylabel(fields[1])

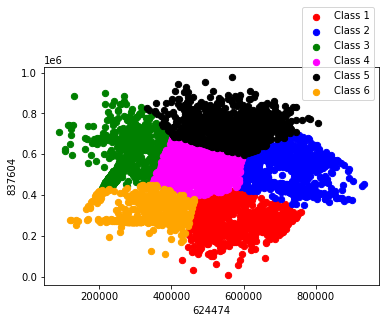
ax.set\_zlabel(fields[2])

ax.legend(loc='upper left', bbox\_to\_anchor=(0.0, 1.3))

plt.show()sds

# Tehtävä 16





## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from mpl\_toolkits.mplot3d import Axes3D

df = pd.read\_csv('data/2dclusters.csv', sep=';', decimal='.')

fields = ['624474', '837604']

X = np.array(df[fields])

inertia = []

for i in range(1,14):

model = KMeans(n\_clusters=i)

model.fit(X)

inertia.append(model.inertia\_)

plt.scatter(np.arange(1,14), inertia)

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

plt.show()

model = KMeans(n\_clusters=6)

model.fit(X)

labels = model.labels\_

df['Label'] = labels

colors = {0:'red', 1:'blue', 2:'green', 3:'magenta', 4:'black', 5:'orange'}

fig = plt.figure()

ax = fig.add\_subplot(111)

for i in range(0,6):

x = df.loc[df['Label'] == i][fields[0]].values

y = df.loc[df['Label'] == i][fields[1]].values

ax.scatter(x, y, marker='o', s=40, color=colors[i], label='Class '+str(i+1))

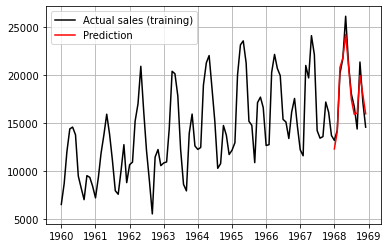
ax.set\_xlabel(fields[0])

ax.set\_ylabel(fields[1])

ax.legend(loc='upper right', bbox\_to\_anchor=(1.0, 1.3))

plt.show()

# Tehtävä 17



## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from sklearn import preprocessing

from sklearn import linear\_model

from sklearn.metrics import mean\_absolute\_error

forecast\_time = 12 # kuukautta

seqlength = 12 # syöteverkon historian pituus

df = pd.read\_csv('data/monthly-car-sales.csv', sep=',', decimal='.')

df['Month'] = pd.to\_datetime(df['Month'])

df['Time'] = df.index

df['SalesLag'] = df['Sales'].shift(1)

df['SalesDiff'] = df.apply(lambda row:

row['Sales']-row['SalesLag'], axis=1)

for i in range(1, seqlength):

df['SalesDiffLag'+str(i)] = df['SalesDiff'].shift(i)

for i in range(1, forecast\_time +1):

df['SalesDiffFut'+str(i)] = df['SalesDiff'].shift(-i)

df\_train = df.iloc[:-2\* forecast\_time]

df\_train.dropna(inplace=True)

df\_test = df.iloc[-2\* forecast\_time:]

# muuttujien valinta ja skaalaus

input\_vars = ['SalesDiff']

for i in range(1, seqlength):

input\_vars.append('SalesDiffLag'+str(i))

output\_vars = []

for i in range(1, forecast\_time +1):

output\_vars.append('SalesDiffFut'+str(i))

scaler = preprocessing.StandardScaler()

scalero = preprocessing.StandardScaler()

X = np.array(df\_train[input\_vars])

X\_scaled = scaler.fit\_transform(X)

X\_scaledLSTM = X\_scaled.reshape(X.shape[0], seqlength, 1)

y = np.array(df\_train[output\_vars])

y\_scaled = scalero.fit\_transform(y)

X\_test = np.array(df\_test[input\_vars])

X\_testscaled = scaler.transform(X\_test)

X\_testscaledLSTM = X\_testscaled.reshape(

X\_test.shape[0], seqlength, 1)

# Trendin mallinnus lineaarisella regressiolla

modelLR = linear\_model.LinearRegression()

XLR = df\_train['Time'].values

XLR = XLR.reshape(-1,1)

yLR = df\_train['Sales'].values

yLR = yLR.reshape(-1,1)

modelLR.fit(XLR, yLR)

XLR\_test = df\_test['Time'].values

XLR\_test = XLR\_test.reshape(-1,1)

df\_test['SalesAvgPred'] = modelLR.predict(XLR\_test)

# trendin kulmakerroin

slope = modelLR.coef\_

# LTSM verkon muodostus ja opetus

modelLSTM = tf.keras.Sequential([

tf.keras.layers.LSTM(24, input\_shape=(seqlength, 1),

return\_sequences=False),

#tf.keras.layers.LSTM(24, return\_sequences=False),

tf.keras.layers.Dense(forecast\_time)

])

modelLSTM.compile(

loss='mse',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

metrics=['mae'])

modelLSTM.fit(X\_scaledLSTM, y\_scaled, epochs=200, batch\_size=seqlength)

# Ennusteen (ennusteDiff + trendi) määritys

ennusteDiff = scalero.inverse\_transform(

modelLSTM.predict(X\_testscaledLSTM[forecast\_time -1].reshape(1,12,1)))

ennuste = np.zeros(13)

ennuste[0] = df\_test['Sales'][df\_test.index[forecast\_time -1]]

for i in range(1,13):

for j in range(1,13):

ennuste[j] = ennuste[j-1]+ennusteDiff[0][j-1]+slope

ennuste = np.array(ennuste[1:])

# ennusteen piirtäminen

df\_pred = df\_test[-12:]

df\_pred['SalesPred'] = ennuste

plt.plot(df['Month'].values, df['Sales'].values, color='black', label='Actual sales (training)')

plt.plot(df\_pred['Month'].values, df\_pred['SalesPred'].values, color='red', label='Prediction')

plt.grid()

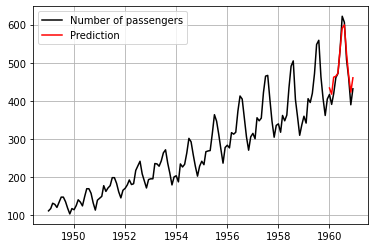
plt.legend()

plt.show()

print(mean\_absolute\_error(df\_pred['Sales'].values,

df\_pred['SalesPred'].values))

# Tehtävä 18



Keskivirhe on 0.233.

## Lähdekoodi

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tensorflow as tf

from sklearn import preprocessing

from sklearn import linear\_model

from sklearn.metrics import mean\_absolute\_error

forecast\_time = 12 # kuukautta

seqlength = 12 # syöteverkon historian pituus

df = pd.read\_csv('data/AirPassengers.csv', sep=',', decimal='.')

df['Month'] = pd.to\_datetime(df['Month'])

df['Time'] = df.index

df['PassengersLag'] = df['Passengers'].shift(1)

df['PassengersDiff'] = df.apply(lambda row:

row['Passengers']-row['PassengersLag'], axis=1)

for i in range(1, seqlength):

df['PassengersDiffLag'+str(i)] = df['PassengersDiff'].shift(i)

for i in range(1, forecast\_time +1):

df['PassengersDiffFut'+str(i)] = df['PassengersDiff'].shift(-i)

df\_train = df.iloc[:-2\* forecast\_time]

df\_train.dropna(inplace=True)

df\_test = df.iloc[-2\* forecast\_time:]

# muuttujien valinta ja skaalaus

input\_vars = ['PassengersDiff']

for i in range(1, seqlength):

input\_vars.append('PassengersDiffLag'+str(i))

output\_vars = []

for i in range(1, forecast\_time +1):

output\_vars.append('PassengersDiffFut'+str(i))

scaler = preprocessing.StandardScaler()

scalero = preprocessing.StandardScaler()

X = np.array(df\_train[input\_vars])

X\_scaled = scaler.fit\_transform(X)

X\_scaledLSTM = X\_scaled.reshape(X.shape[0], seqlength, 1)

y = np.array(df\_train[output\_vars])

y\_scaled = scalero.fit\_transform(y)

X\_test = np.array(df\_test[input\_vars])

X\_testscaled = scaler.transform(X\_test)

X\_testscaledLSTM = X\_testscaled.reshape(

X\_test.shape[0], seqlength, 1)

# Trendin mallinnus lineaarisella regressiolla

modelLR = linear\_model.LinearRegression()

XLR = df\_train['Time'].values

XLR = XLR.reshape(-1,1)

yLR = df\_train['Passengers'].values

yLR = yLR.reshape(-1,1)

modelLR.fit(XLR, yLR)

XLR\_test = df\_test['Time'].values

XLR\_test = XLR\_test.reshape(-1,1)

df\_test['PassengersAvgPred'] = modelLR.predict(XLR\_test)

# trendin kulmakerroin

slope = modelLR.coef\_

# LTSM verkon muodostus ja opetus

modelLSTM = tf.keras.Sequential([

tf.keras.layers.LSTM(24, input\_shape=(seqlength, 1),

return\_sequences=False),

#tf.keras.layers.LSTM(24, return\_sequences=False),

tf.keras.layers.Dense(forecast\_time)

])

modelLSTM.compile(

loss='mse',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

metrics=['mae'])

modelLSTM.fit(X\_scaledLSTM, y\_scaled, epochs=200, batch\_size=seqlength)

# Ennusteen (ennusteDiff + trendi) määritys

ennusteDiff = scalero.inverse\_transform(

modelLSTM.predict(X\_testscaledLSTM[forecast\_time -1].reshape(1,12,1)))

ennuste = np.zeros(13)

ennuste[0] = df\_test['Passengers'][df\_test.index[forecast\_time -1]]

for i in range(1,13):

for j in range(1,13):

ennuste[j] = ennuste[j-1]+ennusteDiff[0][j-1]+slope

ennuste = np.array(ennuste[1:])

# ennusteen piirtäminen

df\_pred = df\_test[-12:]

df\_pred['PassengersPred'] = ennuste

plt.plot(df['Month'].values, df['Passengers'].values, color='black', label='Number of passengers')

plt.plot(df\_pred['Month'].values, df\_pred['PassengersPred'].values, color='red', label='Prediction')

plt.grid()

plt.legend()

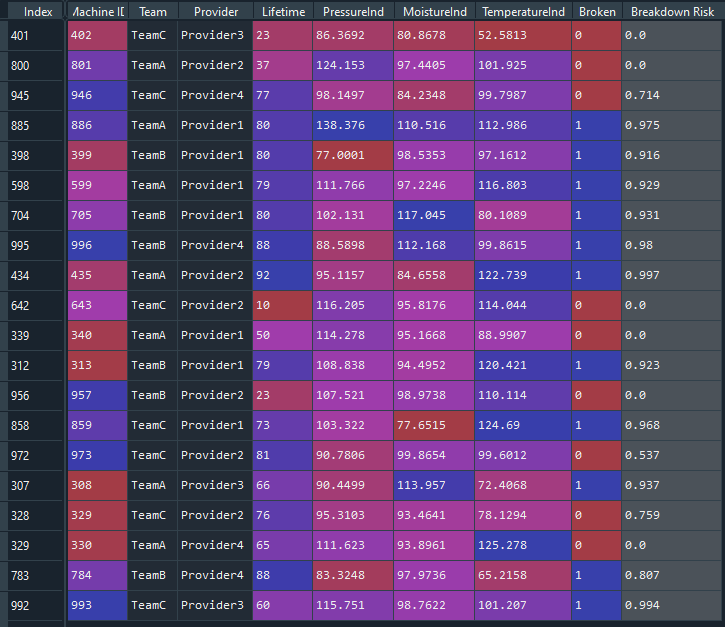
plt.show()

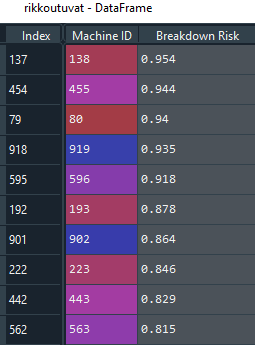
print(mean\_absolute\_error(df\_pred['Passengers'].values,

df\_pred['PassengersPred'].values))

# Tehtävä 19

Malliksi valittiin Kerasin sequential neuroverkkomalli, minkä luokittelutarkkuus asettui 95% yläpuolelle.





## Lähdekoodi

import pandas as pd

import numpy as np

from sklearn import preprocessing

import tensorflow as tf

from tensorflow import keras

df = pd.read\_csv('data/MachineData.csv', sep=';', decimal='.', encoding='utf-8')

teamId = {'TeamA':1, 'TeamB':2, 'TeamC':3}

df['TeamId'] = df['Team'].map(teamId)

df['TeamId'].fillna(-1, inplace=True)

providerId = {'Provider1':1, 'Provider2':2, 'Provider3':3, 'Provider4':4}

df['ProviderId'] = df['Provider'].map(providerId)

df['ProviderId'].fillna(-1, inplace=True)

df\_train = df.sample(n = 200, replace = False)

df\_test = df.drop(df\_train.index)

input\_variables = ['TeamId', 'ProviderId', 'Lifetime', 'PressureInd', 'MoistureInd', 'TemperatureInd']

X = np.array(df[input\_variables])

y = np.array(pd.get\_dummies(df['Broken']))

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = keras.Sequential([

# 1. piilotettu / input kerros

keras.layers.Dense(30, activation=tf.nn.relu, input\_shape=(X\_scaled.shape[1],)),

# 2. piilotetu kerros

keras.layers.Dense(30, activation=tf.nn.relu),

# output kerros -> 2 output luokkaa, softmax tulostaa ko. luokan todennäköisyyden

keras.layers.Dense(2, activation=tf.nn.softmax)

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['categorical\_accuracy'])

model.fit(X\_scaled, y, epochs=20, batch\_size=1)

predictedResults = model.predict(X\_scaled)

model.summary()

roundedResults = np.round(predictedResults, 3)

df['Breakdown Risk'] = roundedResults[:,1]

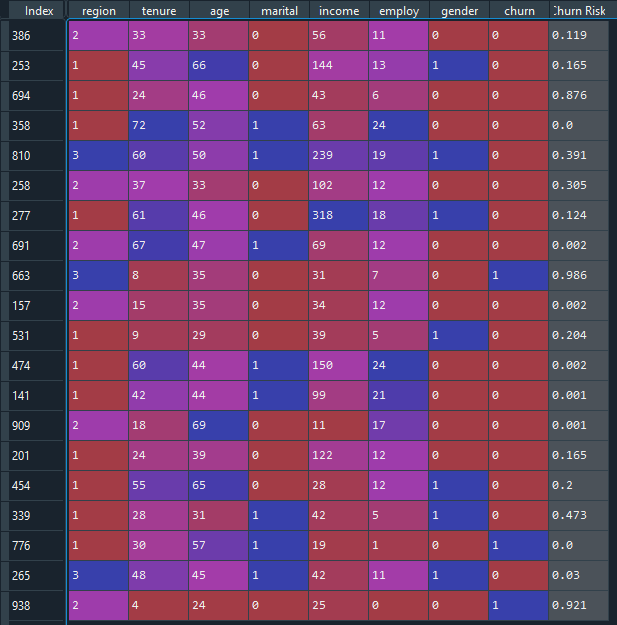
dfResults1 = df.iloc[:, [0,1,2,3,4,5,6,7,10]].sample(20)

dfResults2 = df[df['Broken']==0]

dfResults2.sort\_values(by=['Breakdown Risk'], ascending=False, inplace=True)

rikkoutuvat = dfResults2.iloc[0:10, [0,10]]

# Tehtävä 20: Lähtevien asiakkaiden tunnistaminen



Malliksi valittiin Kerasin sequential neuroverkkomalli. Mallin luokittelutarkkuus oli 95%.

## Lähdekoodi

import pandas as pd

import numpy as np

from sklearn import preprocessing

import tensorflow as tf

from tensorflow import keras

df = pd.read\_csv('data/Telco.csv', sep=';', decimal='.', encoding='utf-8')

df.fillna(0, inplace=True)

input\_variables = df.iloc[:,0:14]

predict\_field = "churn"

df\_train = df.sample(n = 900, replace = False)

df\_test = df.drop(df\_train.index)

# train

X = np.array(df\_train.iloc[:,1:40])

y = np.array(pd.get\_dummies(df\_train[predict\_field]))

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

model = keras.Sequential([

keras.layers.Dense(30, activation=tf.nn.relu, input\_shape=(X\_scaled.shape[1],)),

keras.layers.Dense(30, activation=tf.nn.relu),

keras.layers.Dense(2, activation=tf.nn.softmax)

])

model.compile(optimizer=keras.optimizers.Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['categorical\_accuracy'])

model.fit(X\_scaled, y, epochs=20, batch\_size=1)

# test

X = np.array(df\_test.iloc[:,1:40])

scaler = preprocessing.StandardScaler()

X\_scaled = scaler.fit\_transform(X)

predictedResults = model.predict(X\_scaled)

model.summary()

roundedResults = np.round(predictedResults, 3)

df\_test['Churn Riski'] = roundedResults[:,1]

results\_fields = ['region', 'tenure', 'age', 'marital', 'income', 'employ', 'gender','churn', 'Churn Riski']

df\_results = df\_test[results\_fields].sample(20)s