jamk.fi

# Harjoitukset

Koneoppiminen, Syksy 2020

Tapani Alastalo
M1475

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

# Sisältö

**1 Tehtävä 1 Lineaarinen regressio ..... 3**

1.1 Lähdekoodit ..... 4

**2 Tehtävä 2 ..... 5**

2.1 Lähdekoodit ..... 6

**3 Tehtävä 3 ..... 8**

3.1 Lähdekoodi ..... 8

**4 Tehtävät 4 Neuroverkot ..... 10**

4.1 Lähdekoodi ..... 10

**5 Tehtävä 5 ..... 12**

5.1 Lähdekoodi ..... 12

**6 Tehtävä 6 ..... 14**

6.1 Lähdekoodi ..... 15

6.2 Lähdekoodi ..... 17

**7 Tehtävä 7 ..... 19**

7.1 Lähdekoodi ..... 19

**8 Tehtävä 8 ..... 21**

8.1 Lähdekoodi ..... 21

**9 Tehtävä 9 ..... 23**

9.1 Lähdekoodi ..... 24

**10 Tehtävä 10 ..... 26**

10.1 Lähdekoodi ..... 27

**11 Tehtävä 11 ..... 28**

11.1 Lähdekoodi ..... 30

**12 Tehtävä 12 ..... 31**

12.1 Lähdekoodi ..... 33

**13 Tehtävä 13 ..... 34**

13.1 Lähdekoodi ..... 34

**14 Tehtävä 14 ..... 34**

14.1 Lähdekoodi ..... 34

**15 Tehtävä 15 ..... 36**

15.1 Lähdekoodi ..... 37

**16 Tehtävä 16 ..... 38**

16.1 Lähdekoodi ..... 39

**17 Tehtävä 17 ..... 40**

17.1 Lähdekoodi ..... 40

**18 Tehtävä 18 ..... 43**

18.1 Lähdekoodi ..... 43

**19 Tehtävä 19 ..... 46**

19.1 Lähdekoodi ..... 47

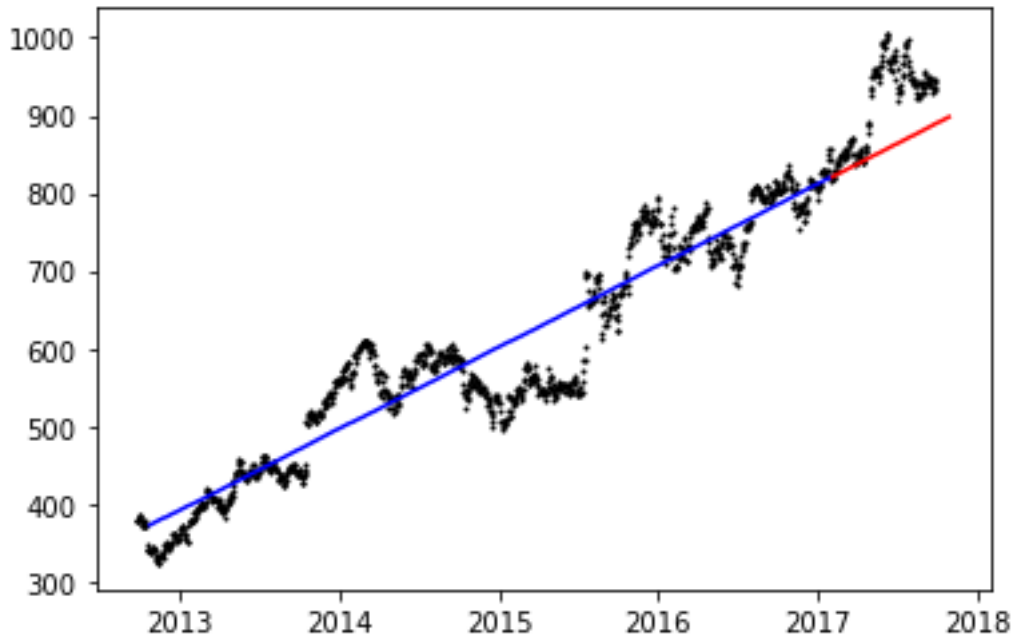
**20 Tehtävä 20: Lähtevien asiakkaiden tunnistaminen ..... 49**

20.1 Lähdekoodi ..... 49

jamk.fi

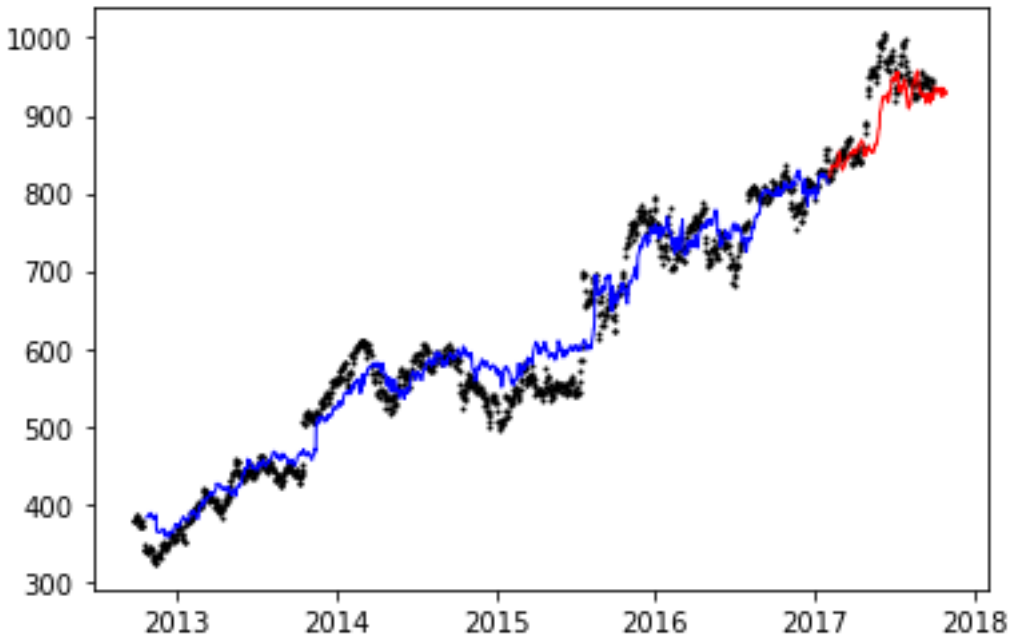
# 1 Tehtävä 1 Lineaarinen regressio

a)




Ennusteen keskivirhe testidatassa on 70

b)




Ennusteen keskivirhe testidatassa on 35

## 1.1 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_valida-
tion['CloseFuture'], df_validation['Ennuste']))
```

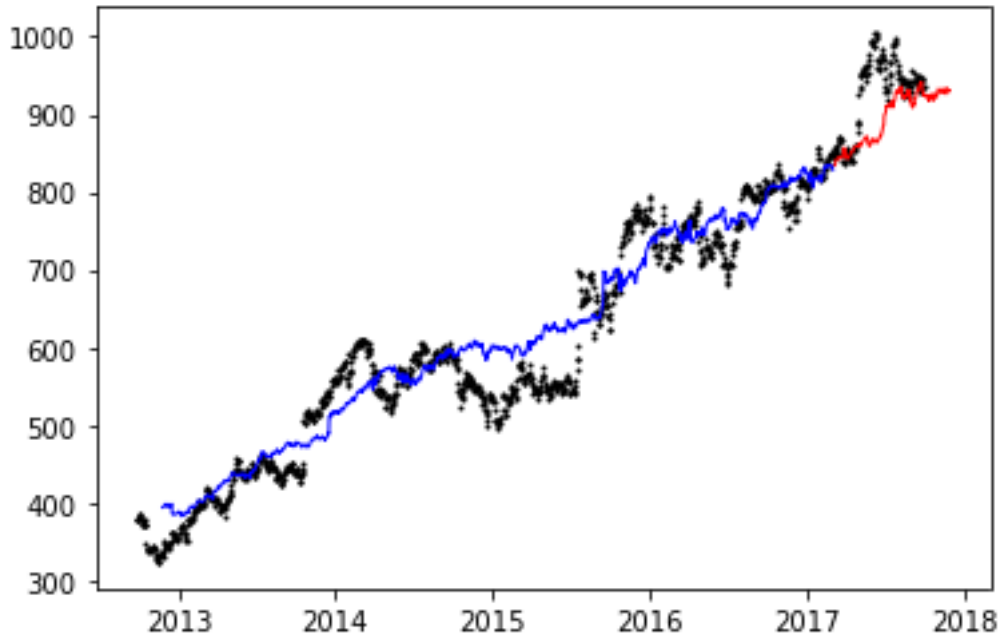
## 2 Tehtävä 2

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

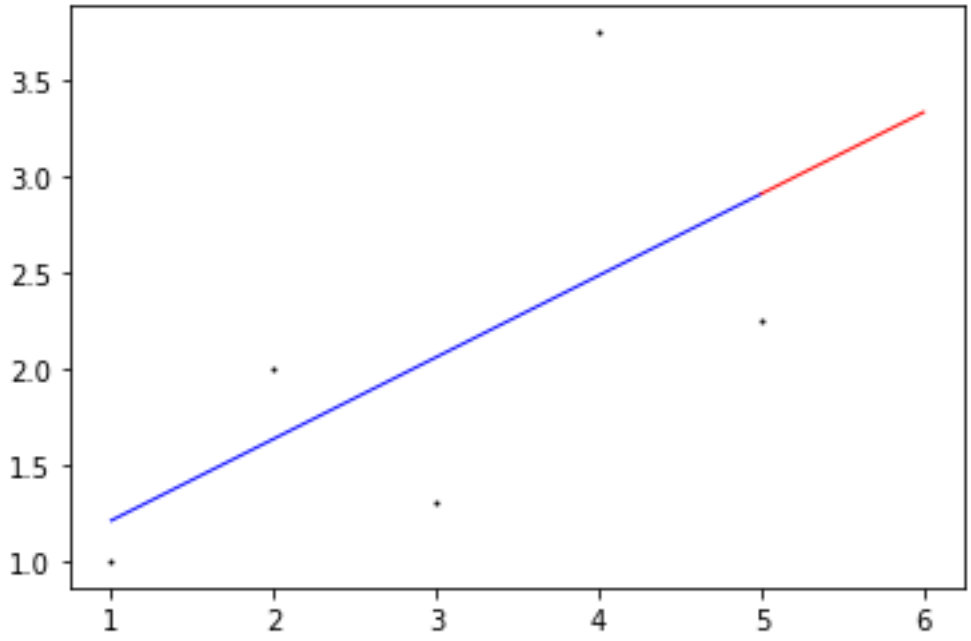
Ennusteen keskivirhe testidatassa on 60

## 2.1 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['En-
nuste'].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_er-
ror(df_train_validation['CloseFuture'], df_train_validation['Ennuste']))
print("Ennusteen keskivirhe test datassa on %.f" % mean_absolute_er-

ror(df_test_validation['CloseFuture'], df_test_validation['Ennuste']))
```

# 3 Tehtävä 3



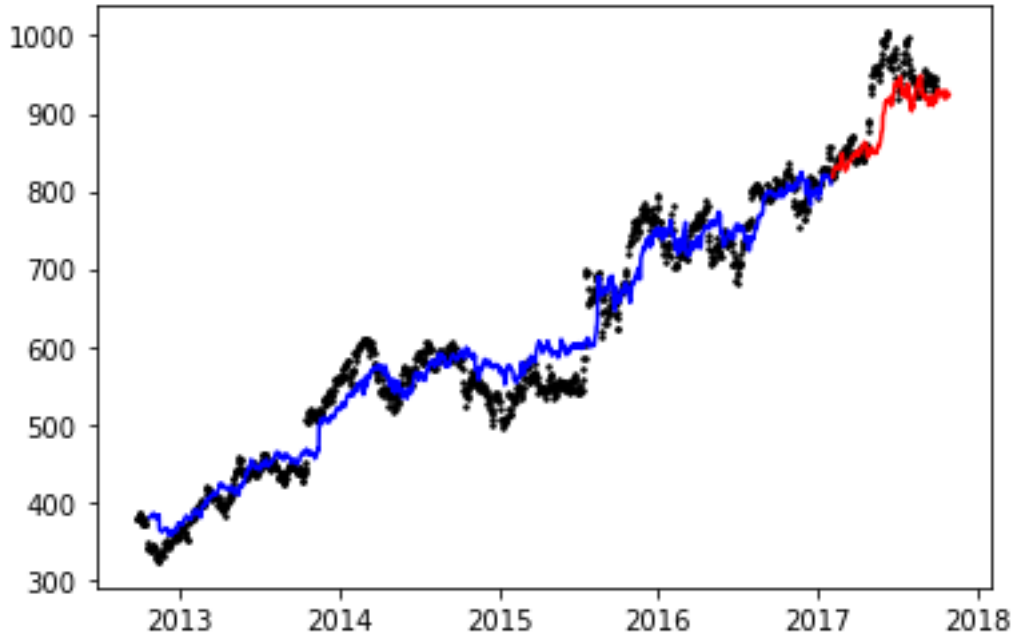

| Index | X | Y | Ennuste |
|---|---|---|---|
| 0 | 1 | 1 | 1.21 |
| 1 | 2 | 2 | 1.635 |
| 2 | 3 | 1.3 | 2.06 |
| 3 | 4 | 3.75 | 2.485 |
| 4 | 5 | 2.25 | 2.91 |
| 5 | 6 | nan | 3.335 |

## 3.1 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)
```

# 4 Tehtävät 4 Neuroverkot




Ennusteen keskivirhe opetusdatassa on 30

Ennusteen keskivirhe testidatassa on 40

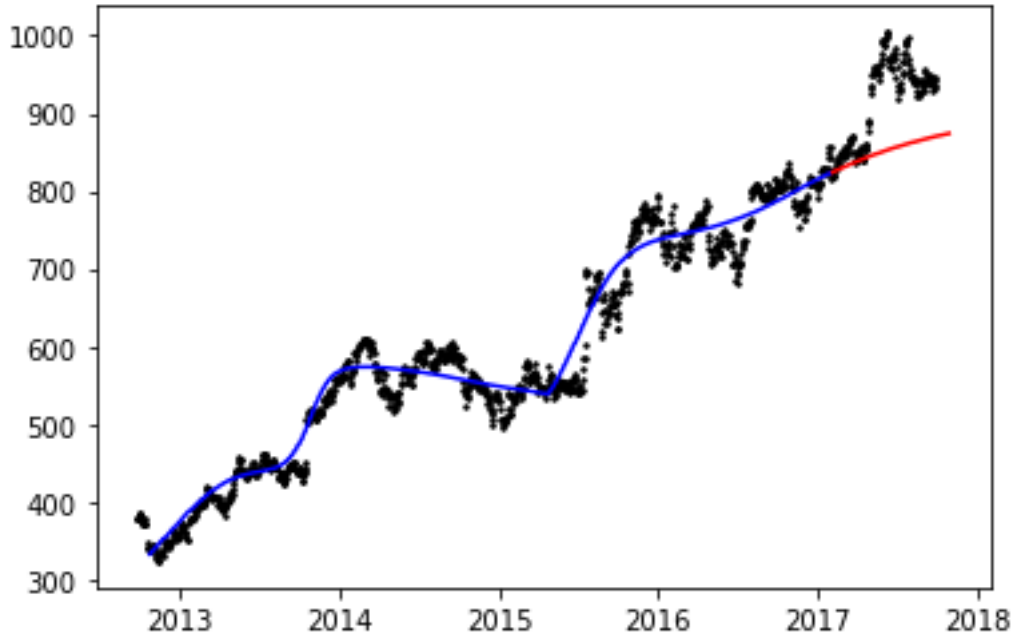
## 4.1 Lähdekoodi

```
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['En-
nuste']))
print("Ennusteen keskivirhe testidatassa on %.f" %
    mean_absolute_error(df_test_validation['CloseFuture'], df_test_validation['En-
nuste']))
```

# 5 Tehtävä 5




Ennusteen keskivirhe opetusdatassa on 19

Ennusteen keskivirhe testidatassa on 74

## 5.1 Lähdekoodi

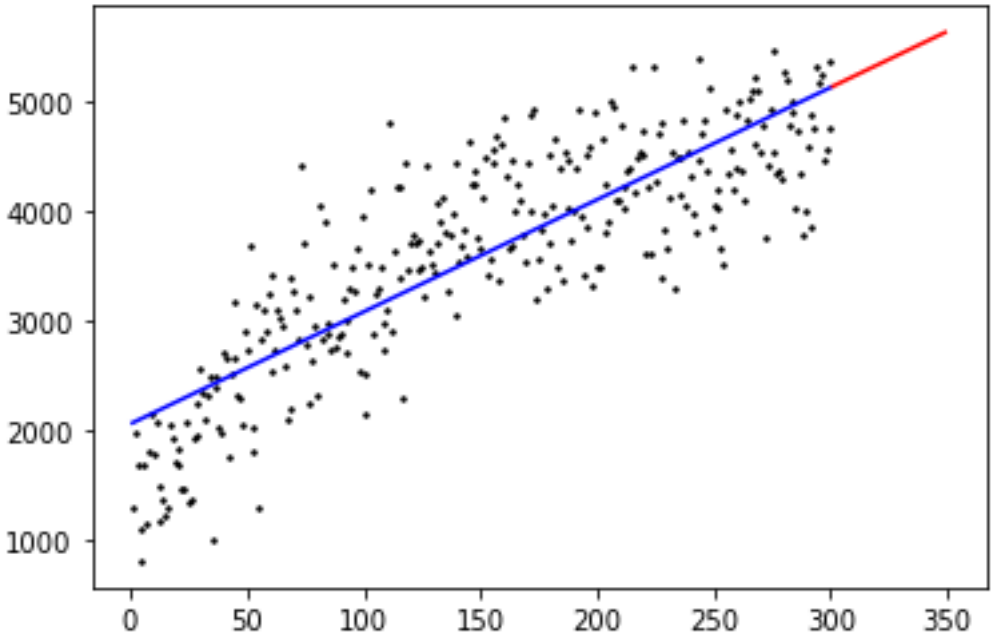
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error
import tensorflow as tf
```

```
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['En-
nuste']))
print("Ennusteen keskivirhe testidatassa on %.f" %
    mean_absolute_error(df_test_validation['CloseFuture'], df_test_validation['En-
nuste']))ss
```

# 6 Tehtävä 6

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



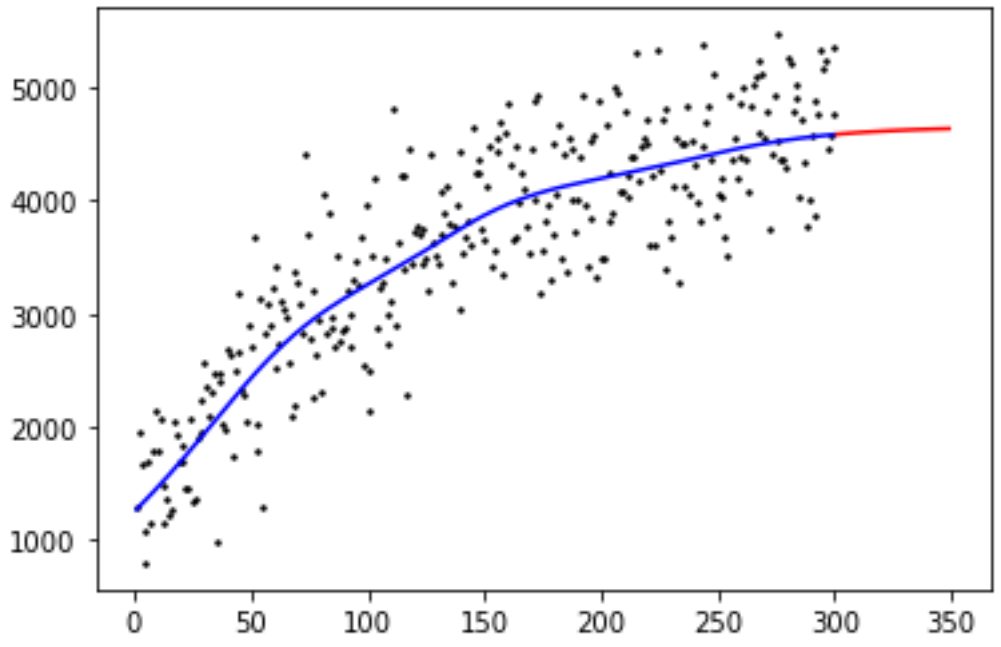

Ennusteen keskivirhe opetusdatassa on 452

## 6.1 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['En-
nuste']))
```

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




Ennusteen keskivirhe opetusdatassa on 405

Ennusteen keskivirhe testidatassa on 407

## 6.2 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['En-
nuste']))
```

# 7 Tehtävä 7

| Index | fruit_name | fruit_subtype | mass | width | height | olor_scor | fruit_code | LRennuste | VMennust | NNennust |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | apple | granny_smith | 192 | 8.4 | 7.3 | 0.55 | 0 | 0 | 0 | 0 |
| 1 | apple | granny_smith | 180 | 8 | 6.8 | 0.59 | 0 | 0 | 0 | 0 |
| 2 | apple | granny_smith | 176 | 7.4 | 7.2 | 0.6 | 0 | 3 | 0 | 0 |
| 3 | mandarin | mandarin | 86 | 6.2 | 4.7 | 0.8 | 2 | 2 | 2 | 2 |
| 4 | mandarin | mandarin | 84 | 6 | 4.6 | 0.79 | 2 | 2 | 2 | 2 |
| 5 | mandarin | mandarin | 80 | 5.8 | 4.3 | 0.77 | 2 | 2 | 2 | 2 |
| 6 | mandarin | mandarin | 80 | 5.9 | 4.3 | 0.81 | 2 | 2 | 2 | 2 |
| 7 | mandarin | mandarin | 76 | 5.8 | 4 | 0.81 | 2 | 2 | 2 | 2 |
| 8 | apple | braeburn | 178 | 7.1 | 7.8 | 0.92 | 0 | 3 | 0 | 0 |
| 9 | apple | braeburn | 172 | 7.4 | 7 | 0.89 | 0 | 0 | 0 | 0 |
| 10 | apple | braeburn | 166 | 6.9 | 7.3 | 0.93 | 0 | 0 | 0 | 0 |
| 11 | apple | braeburn | 172 | 7.1 | 7.6 | 0.92 | 0 | 0 | 0 | 0 |
| 12 | apple | braeburn | 154 | 7 | 7.1 | 0.88 | 0 | 0 | 0 | 0 |
| 13 | apple | golden_delicious | 164 | 7.3 | 7.7 | 0.7 | 0 | 3 | 3 | 0 |
| 14 | apple | golden_delicious | 152 | 7.6 | 7.3 | 0.69 | 0 | 0 | 0 | 0 |
| 15 | apple | golden_delicious | 156 | 7.7 | 7.1 | 0.69 | 0 | 0 | 0 | 0 |

Ennusteen keskivirhe logistisella regressiolla on 0.881.

Ennusteen keskivirhe SVM luokittelulla on 0.966.

Ennusteen keskivirhe K-Lähimmännaapurin luokittelulla on 0.983.

## 7.1 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
```

# 8 Tehtävä 8

| Index | fruit_name | fruit_subtype | mass | width | height | olor_scor | Ennuste |
|---|---|---|---|---|---|---|---|
| 0 | apple | granny_smith | 192 | 8.4 | 7.3 | 0.55 | 0 |
| 1 | apple | granny_smith | 180 | 8 | 6.8 | 0.59 | 0 |
| 2 | apple | granny_smith | 176 | 7.4 | 7.2 | 0.6 | 0 |
| 3 | mandarin | mandarin | 86 | 6.2 | 4.7 | 0.8 | 2 |
| 4 | mandarin | mandarin | 84 | 6 | 4.6 | 0.79 | 2 |
| 5 | mandarin | mandarin | 80 | 5.8 | 4.3 | 0.77 | 2 |
| 6 | mandarin | mandarin | 80 | 5.9 | 4.3 | 0.81 | 2 |
| 7 | mandarin | mandarin | 76 | 5.8 | 4 | 0.81 | 2 |
| 8 | apple | braeburn | 178 | 7.1 | 7.8 | 0.92 | 0 |
| 9 | apple | braeburn | 172 | 7.4 | 7 | 0.89 | 0 |
| 10 | apple | braeburn | 166 | 6.9 | 7.3 | 0.93 | 0 |
| 11 | apple | braeburn | 172 | 7.1 | 7.6 | 0.92 | 0 |
| 12 | apple | braeburn | 154 | 7 | 7.1 | 0.88 | 0 |
| 13 | apple | golden_delicious | 164 | 7.3 | 7.7 | 0.7 | 0 |
| 14 | apple | golden_delicious | 152 | 7.6 | 7.3 | 0.69 | 0 |
| 15 | apple | golden_delicious | 156 | 7.7 | 7.1 | 0.69 | 0 |

Osumatarkkuus = 100 % harjoitusdatalla, kun käytetään 20 opetuskertaa.

## 8.1 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']))
```

jamk.fi

```
# 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
```

# 9 Tehtävä 9

| Index | assengerI | Survived | LRennuste | vMennust | NNennust |
|---|---|---|---|---|---|
| 165 | 166 | 1 | 0 | 0 | 1 |
| 331 | 332 | 0 | 1 | 0 | 0 |
| 212 | 213 | 0 | 0 | 0 | 0 |
| 764 | 765 | 0 | 0 | 0 | 0 |
| 849 | 850 | 1 | 1 | 1 | 1 |
| 160 | 161 | 0 | 0 | 0 | 0 |
| 788 | 789 | 1 | 0 | 0 | 0 |
| 8 | 9 | 1 | 1 | 1 | 1 |
| 593 | 594 | 0 | 1 | 1 | 1 |
| 147 | 148 | 0 | 1 | 0 | 0 |
| 641 | 642 | 1 | 1 | 1 | 1 |
| 521 | 522 | 0 | 0 | 0 | 0 |
| 815 | 816 | 0 | 1 | 0 | 0 |
| 796 | 797 | 1 | 1 | 1 | 1 |
| 547 | 548 | 1 | 0 | 0 | 0 |
| 92 | 93 | 0 | 0 | 0 | 0 |
| 860 | 861 | 0 | 0 | 0 | 0 |
| 657 | 658 | 0 | 1 | 1 | 1 |
| 121 | 122 | 0 | 0 | 0 | 0 |
| 564 | 565 | 0 | 1 | 1 | 1 |

Ennusteen keskivirhe logistisella regressiolla on 0.781.

Ennusteen keskivirhe SVM luokittelulla on 0.800.

Ennusteen keskivirhe K-Lähimmännaapurin luokittelulla on 0.781.

## 9.1 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)
```

## 10 Tehtävä 10

| Index | assengerI | Survived | Ennuste |
|---|---|---|---|
| 669 | 670 | 1 | 1 |
| 40 | 41 | 0 | 1 |
| 63 | 64 | 0 | 0 |
| 232 | 233 | 0 | 0 |
| 536 | 537 | 0 | 1 |
| 141 | 142 | 1 | 0 |
| 57 | 58 | 0 | 0 |
| 100 | 101 | 0 | 0 |
| 212 | 213 | 0 | 0 |
| 14 | 15 | 0 | 0 |
| 98 | 99 | 1 | 1 |
| 585 | 586 | 1 | 1 |
| 97 | 98 | 1 | 0 |
| 433 | 434 | 0 | 0 |
| 421 | 422 | 0 | 0 |
| 5 | 6 | 0 | 0 |
| 504 | 505 | 1 | 1 |
| 571 | 572 | 1 | 1 |
| 275 | 276 | 1 | 1 |
| 862 | 863 | 1 | 1 |

Ennusteen tarkkuus on 0.855.

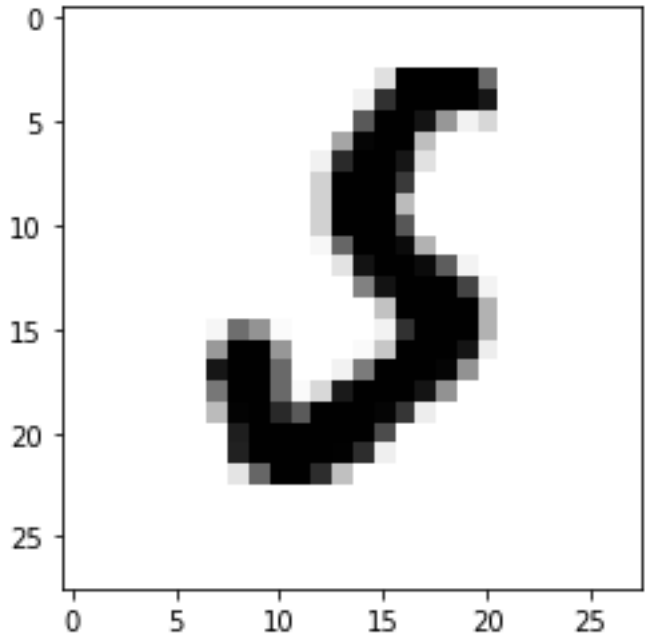
## 10.1 Lähdekoodi

```
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)
```

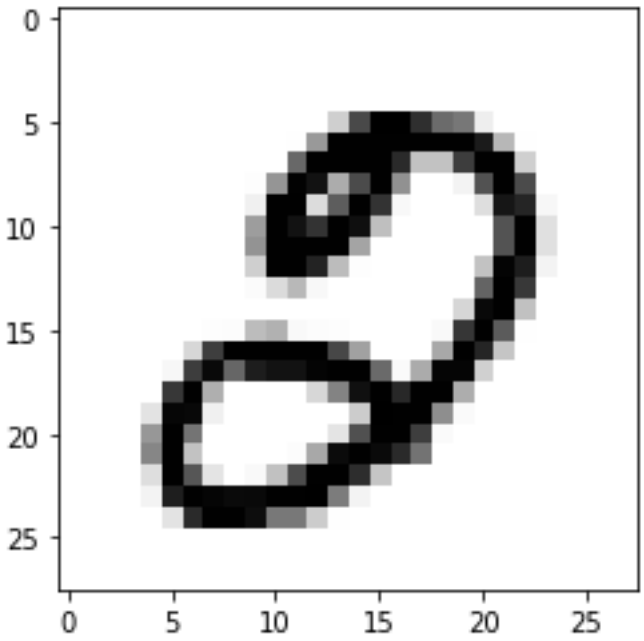
```
# test
X = np.array(df_test[input_variables])
#y = np.array(pd.get_dummies(df_test['Survived']))
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X)

# 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
```

## 11 Tehtävä 11

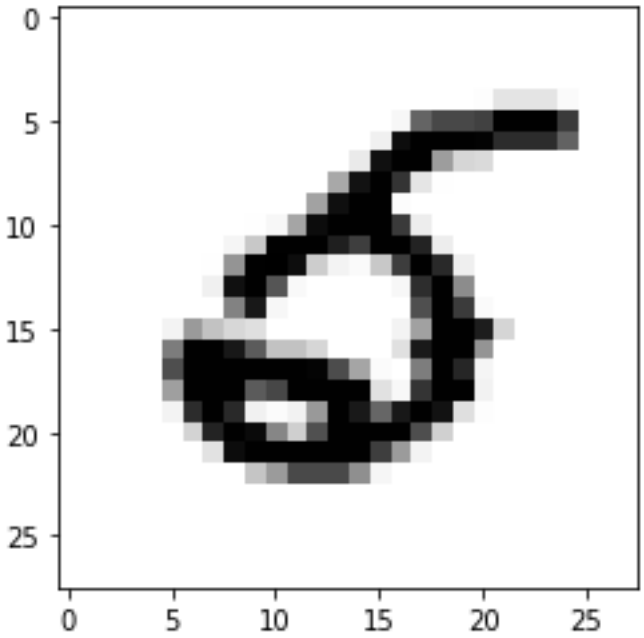
Kategorointitarkkuus opetusdatassa saavutti 99.5% tarkkuuden ja testidatassa 98.0 % tarkkuuden.




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.



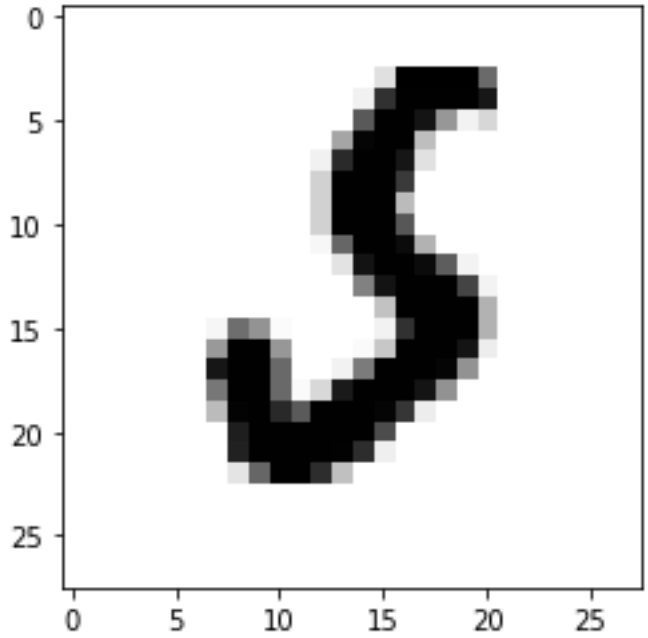

Malli tunnistaa yllä olevan kuvan 50 % todennäköisyydellä numeroksi 3, 30 % todennäköisyydellä numeroksi 6 ja 19 % todennäköisyydellä numeroksi 5. Ihmissilmä näkee tuossa numeroa 5 ja 6.

## 11.1 Lähdekoodi

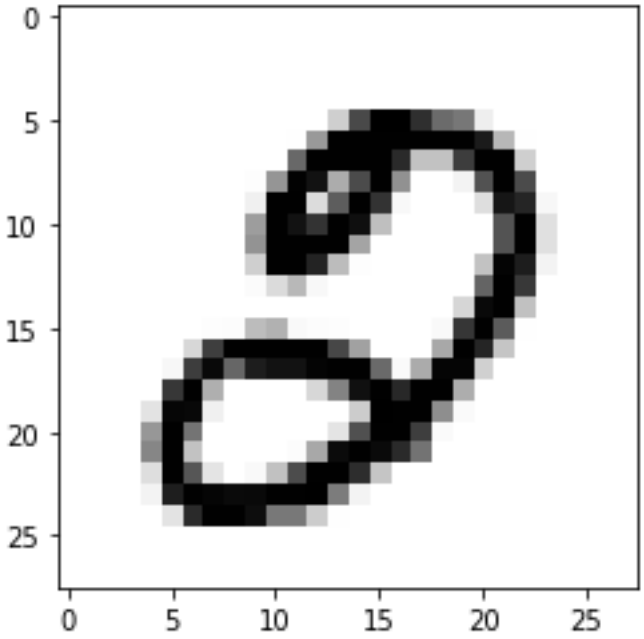
```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

(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')
```

# 12 Tehtävä 12

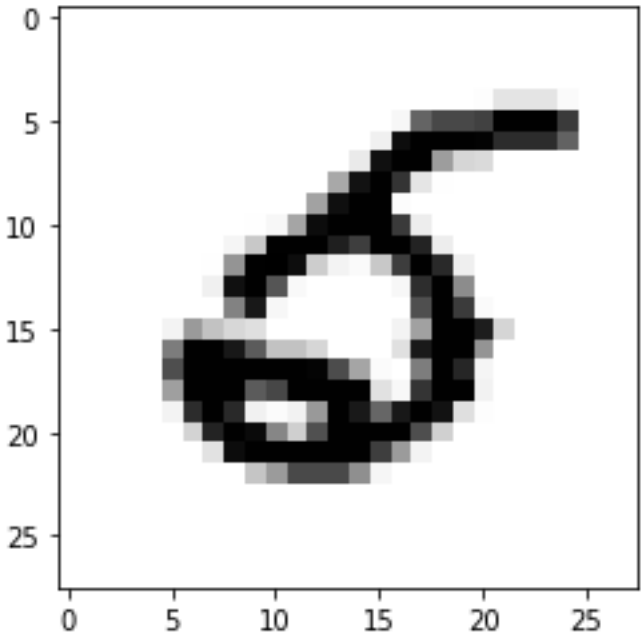
Kategorointitarkkuus opetusdatassa saavutti 99.6% tarkkuuden ja testidatassa 99.1 % tarkkuuden.




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.

## 12.1 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')
```

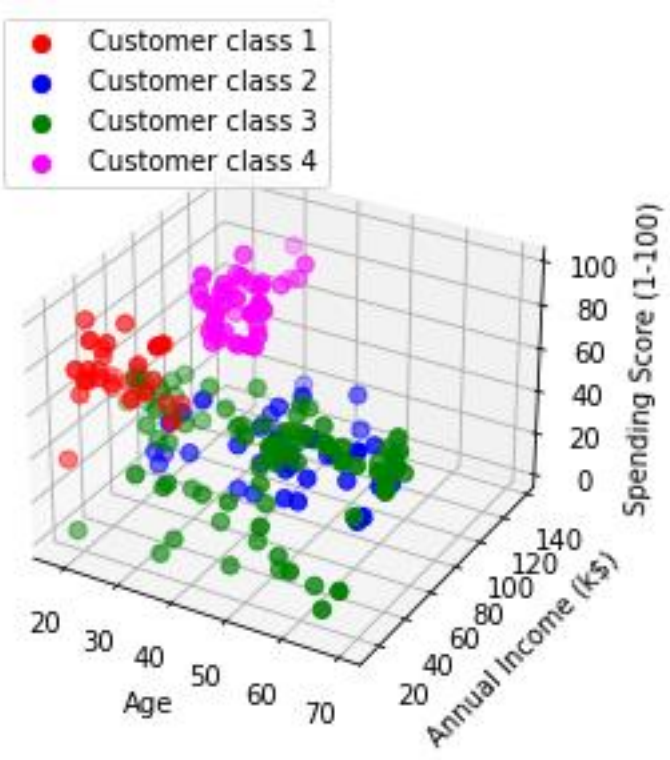
# 13 Tehtävä 13

Sd

## 13.1 Lähdekoodi

sds

# 14 Tehtävä 14



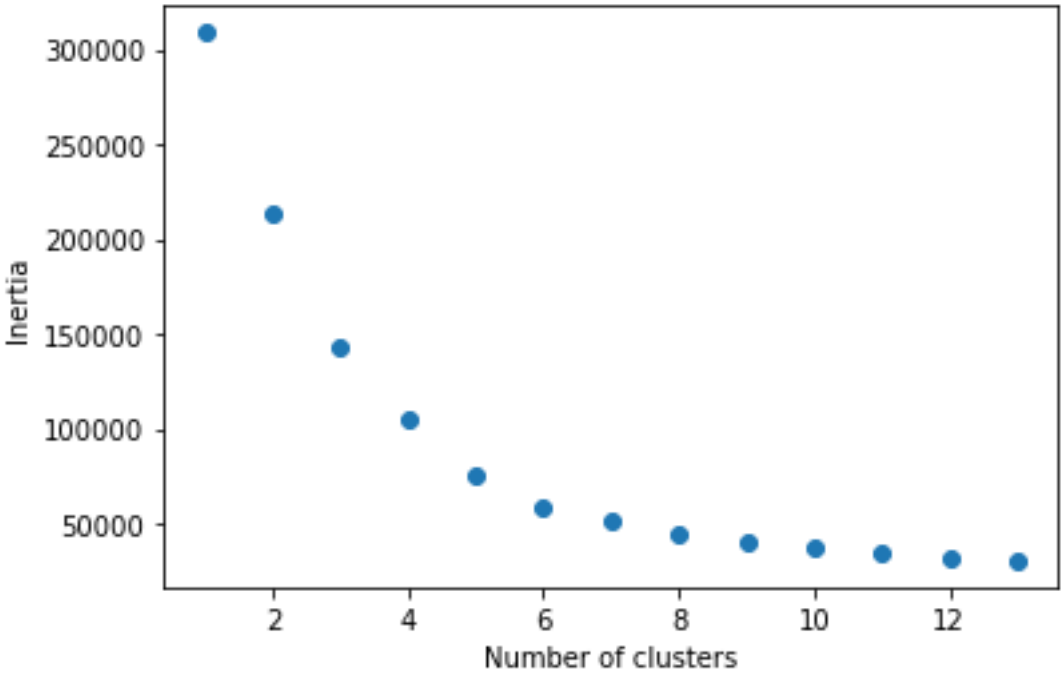

## 14.1 Lähdekoodi

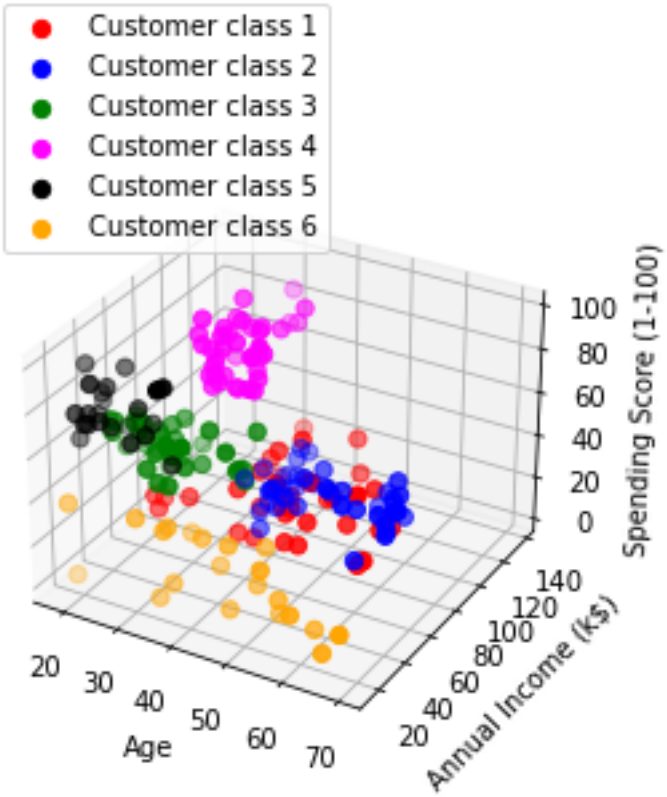
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from mpl_toolkits.mplot3d import Axes3D
```

```
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()
```

jamk.fi

# 15 Tehtävä 15



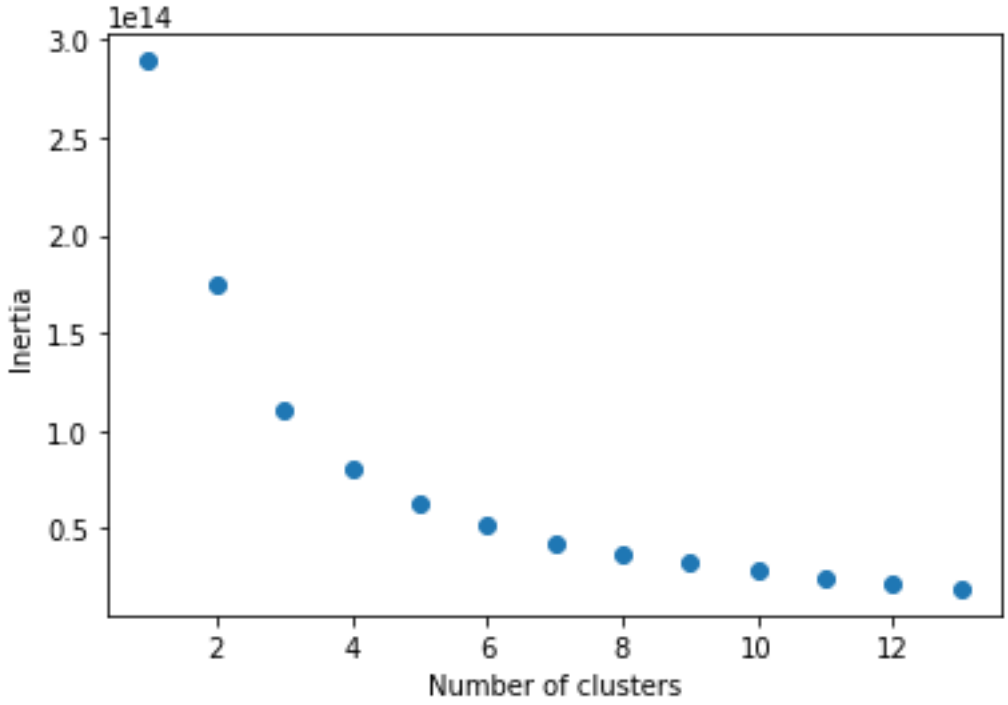



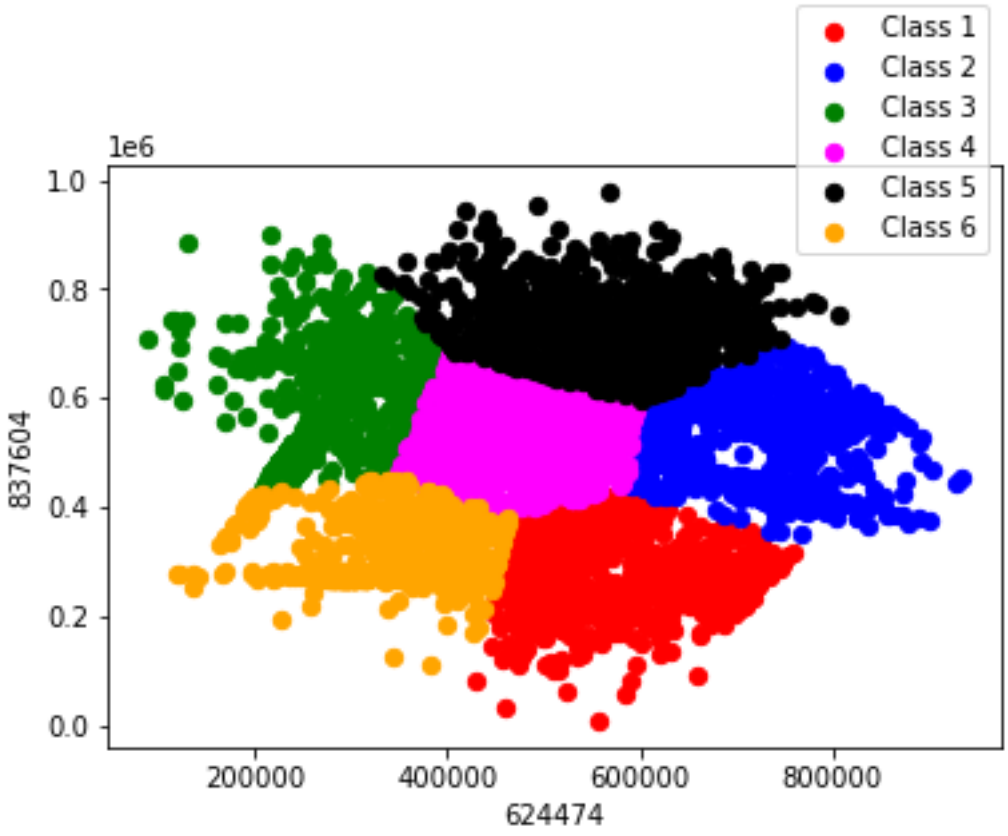
## 15.1 Lähdekoodi

```
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
```

jamk.fi

# 16 Tehtävä 16



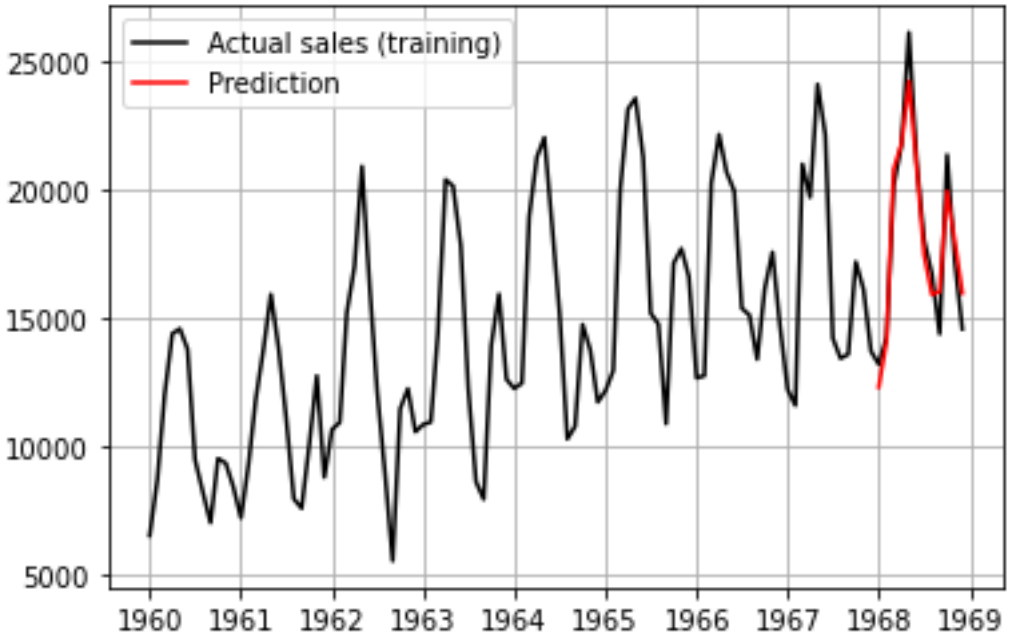



## 16.1 Lähdekoodi

```
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()
```

jamk.fi

# 17 Tehtävä 17




## 17.1 Lähdekoodi

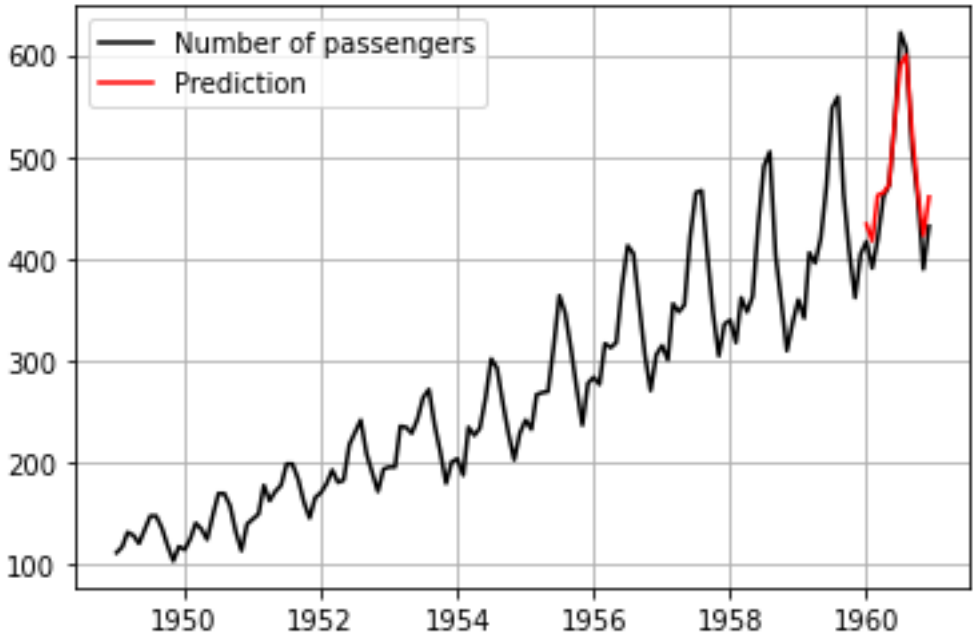
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn import preprocessing
from sklearn import linear_model
from sklearn.metrics import mean_absolute_error

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 (train-
ing)')
plt.plot(df_pred['Month'].values, df_pred['SalesPred'].values, color='red', label='Pre-
diction')

plt.grid()
plt.legend()
plt.show()

print(mean_absolute_error(df_pred['Sales'].values,
                          df_pred['SalesPred'].values))
```

jamk.fi

# 18 Tehtävä 18




Keskivirhe on 0.233.

## 18.1 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', la-
bel='Prediction')

plt.grid()
plt.legend()
plt.show()

print(mean_absolute_error(df_pred['Passengers'].values,
                          df_pred['PassengersPred'].values))
```

jamk.fi

# 19 Tehtävä 19

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

| Index | Machine ID | Team | Provider | Lifetime | PressureInd | MoistureInd | TemperatureInd | Broken | Breakdown Risk |
|---|---|---|---|---|---|---|---|---|---|
| 401 | 402 | TeamC | Provider3 | 23 | 86.3692 | 80.8678 | 52.5813 | 0 | 0.0 |
| 800 | 801 | TeamA | Provider2 | 37 | 124.153 | 97.4405 | 101.925 | 0 | 0.0 |
| 945 | 946 | TeamC | Provider4 | 77 | 98.1497 | 84.2348 | 99.7987 | 0 | 0.714 |
| 885 | 886 | TeamA | Provider1 | 80 | 138.376 | 110.516 | 112.986 | 1 | 0.975 |
| 398 | 399 | TeamB | Provider1 | 80 | 77.0001 | 98.5353 | 97.1612 | 1 | 0.916 |
| 598 | 599 | TeamA | Provider1 | 79 | 111.766 | 97.2246 | 116.803 | 1 | 0.929 |
| 704 | 705 | TeamB | Provider1 | 80 | 102.131 | 117.045 | 80.1089 | 1 | 0.931 |
| 995 | 996 | TeamB | Provider4 | 88 | 88.5898 | 112.168 | 99.8615 | 1 | 0.98 |
| 434 | 435 | TeamA | Provider2 | 92 | 95.1157 | 84.6558 | 122.739 | 1 | 0.997 |
| 642 | 643 | TeamC | Provider2 | 10 | 116.205 | 95.8176 | 114.044 | 0 | 0.0 |
| 339 | 340 | TeamA | Provider1 | 50 | 114.278 | 95.1668 | 88.9907 | 0 | 0.0 |
| 312 | 313 | TeamB | Provider1 | 79 | 108.838 | 94.4952 | 120.421 | 1 | 0.923 |
| 956 | 957 | TeamB | Provider2 | 23 | 107.521 | 98.9738 | 110.114 | 0 | 0.0 |
| 858 | 859 | TeamC | Provider1 | 73 | 103.322 | 77.6515 | 124.69 | 1 | 0.968 |
| 972 | 973 | TeamC | Provider2 | 81 | 90.7806 | 99.8654 | 99.6012 | 0 | 0.537 |
| 307 | 308 | TeamA | Provider3 | 66 | 90.4499 | 113.957 | 72.4068 | 1 | 0.937 |
| 328 | 329 | TeamC | Provider2 | 76 | 95.3103 | 93.4641 | 78.1294 | 0 | 0.759 |
| 329 | 330 | TeamA | Provider4 | 65 | 111.623 | 93.8961 | 125.278 | 0 | 0.0 |
| 783 | 784 | TeamB | Provider4 | 88 | 83.3248 | 97.9736 | 65.2158 | 1 | 0.807 |
| 992 | 993 | TeamC | Provider3 | 60 | 115.751 | 98.7622 | 101.207 | 1 | 0.994 |

rikkoutuvat - DataFrame

| Index | Machine ID | Breakdown Risk |
|---|---|---|
| 137 | 138 | 0.954 |
| 454 | 455 | 0.944 |
| 79 | 80 | 0.94 |
| 918 | 919 | 0.935 |
| 595 | 596 | 0.918 |
| 192 | 193 | 0.878 |
| 901 | 902 | 0.864 |
| 222 | 223 | 0.846 |
| 442 | 443 | 0.829 |
| 562 | 563 | 0.815 |

## 19.1 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]]
```

jamk.fi

# 20 Tehtävä 20: Lähtevien asiakkaiden tunnistaminen

| Index | region | tenure | age | marital | income | employ | gender | churn | Churn Risk |
|---|---|---|---|---|---|---|---|---|---|
| 386 | 2 | 33 | 33 | 0 | 56 | 11 | 0 | 0 | 0.119 |
| 253 | 1 | 45 | 66 | 0 | 144 | 13 | 1 | 0 | 0.165 |
| 694 | 1 | 24 | 46 | 0 | 43 | 6 | 0 | 0 | 0.876 |
| 358 | 1 | 72 | 52 | 1 | 63 | 24 | 0 | 0 | 0.0 |
| 810 | 3 | 60 | 50 | 1 | 239 | 19 | 1 | 0 | 0.391 |
| 258 | 2 | 37 | 33 | 0 | 102 | 12 | 0 | 0 | 0.305 |
| 277 | 1 | 61 | 46 | 0 | 318 | 18 | 1 | 0 | 0.124 |
| 691 | 2 | 67 | 47 | 1 | 69 | 12 | 0 | 0 | 0.002 |
| 663 | 3 | 8 | 35 | 0 | 31 | 7 | 0 | 1 | 0.986 |
| 157 | 2 | 15 | 35 | 0 | 34 | 12 | 0 | 0 | 0.002 |
| 531 | 1 | 9 | 29 | 0 | 39 | 5 | 1 | 0 | 0.204 |
| 474 | 1 | 60 | 44 | 1 | 150 | 24 | 0 | 0 | 0.002 |
| 141 | 1 | 42 | 44 | 1 | 99 | 21 | 0 | 0 | 0.001 |
| 909 | 2 | 18 | 69 | 0 | 11 | 17 | 0 | 0 | 0.001 |
| 201 | 1 | 24 | 39 | 0 | 122 | 12 | 0 | 0 | 0.165 |
| 454 | 1 | 55 | 65 | 0 | 28 | 12 | 1 | 0 | 0.2 |
| 339 | 1 | 28 | 31 | 1 | 42 | 5 | 1 | 0 | 0.473 |
| 776 | 1 | 30 | 57 | 1 | 19 | 1 | 0 | 1 | 0.0 |
| 265 | 3 | 48 | 45 | 1 | 42 | 11 | 1 | 0 | 0.03 |
| 938 | 2 | 4 | 24 | 0 | 25 | 0 | 0 | 1 | 0.921 |

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

## 20.1 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
```