



# Harjoitukset

Koneoppiminen, Syksy 2020

Tapani Alastalo  
M1475

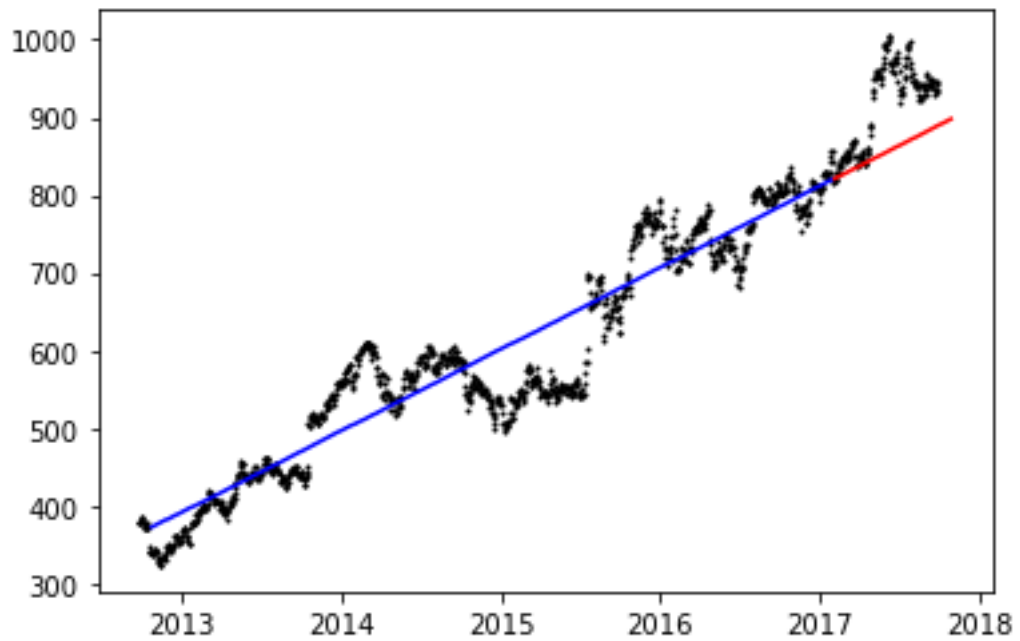
## Sisältö

<b>1</b>	<b>Tehtävä 1 Lineaarinen regressio.....</b>	<b>3</b>
1.1	Lähdekoodit.....	4
<b>2</b>	<b>Tehtävä 2 .....</b>	<b>5</b>
2.1	Lähdekoodit.....	6
<b>3</b>	<b>Tehtävä 3 .....</b>	<b>8</b>
3.1	Lähdekoodi .....	8
<b>4</b>	<b>Tehtävät 4 Neuroverkot.....</b>	<b>10</b>
4.1	Lähdekoodi .....	10
<b>5</b>	<b>Tehtävä 5 .....</b>	<b>12</b>
5.1	Lähdekoodi .....	12
<b>6</b>	<b>Tehtävä 6 .....</b>	<b>14</b>
6.1	Lähdekoodi .....	15
6.2	Lähdekoodi .....	17
<b>7</b>	<b>Tehtävä 7 .....</b>	<b>19</b>
7.1	Lähdekoodi .....	19
<b>8</b>	<b>Tehtävä 8 .....</b>	<b>21</b>
8.1	Lähdekoodi .....	21
<b>9</b>	<b>Tehtävä 9 .....</b>	<b>23</b>
9.1	Lähdekoodi .....	24
<b>10</b>	<b>Tehtävä 10.....</b>	<b>26</b>
10.1	Lähdekoodi .....	27
<b>11</b>	<b>Tehtävä 11.....</b>	<b>28</b>
11.1	Lähdekoodi .....	30
<b>12</b>	<b>Tehtävä 12.....</b>	<b>31</b>
12.1	Lähdekoodi .....	33

<b>13</b>	<b>Tehtävä 13.....</b>	<b>34</b>
13.1	Lähdekoodi .....	34
<b>14</b>	<b>Tehtävä 14.....</b>	<b>34</b>
14.1	Lähdekoodi .....	34
<b>15</b>	<b>Tehtävä 15.....</b>	<b>36</b>
15.1	Lähdekoodi .....	37
<b>16</b>	<b>Tehtävä 16.....</b>	<b>38</b>
16.1	Lähdekoodi .....	39
<b>17</b>	<b>Tehtävä 17.....</b>	<b>40</b>
17.1	Lähdekoodi .....	40
<b>18</b>	<b>Tehtävä 18.....</b>	<b>43</b>
18.1	Lähdekoodi .....	43
<b>19</b>	<b>Tehtävä 19.....</b>	<b>46</b>
19.1	Lähdekoodi .....	46
<b>20</b>	<b>Tehtävä 20: Lähtevien asiakkaiden tunnistaminen .....</b>	<b>46</b>
20.1	Lähdekoodi .....	46

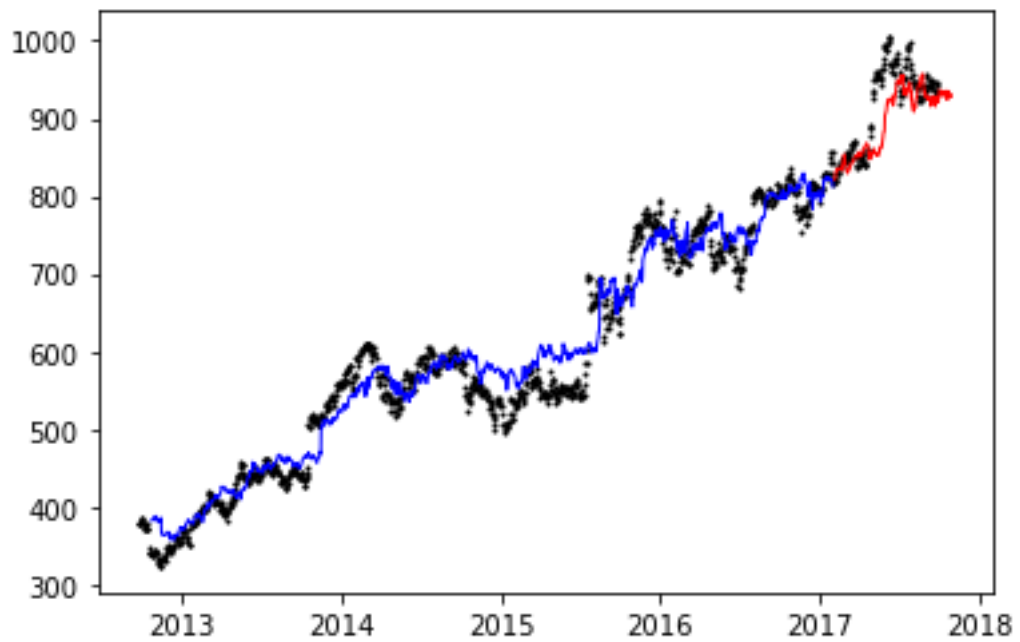
## 1 Tehtävä 1 Lineaarinen regressio

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')
```

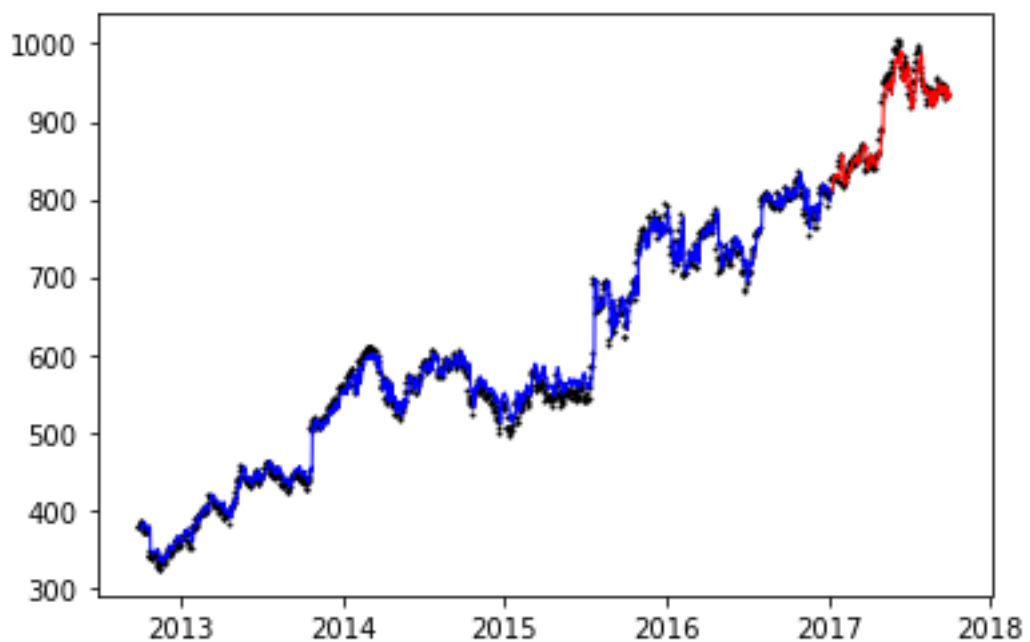
```
plt.show()
```

```
df_validation = df_test.dropna()
```

```
print("Ennusteen keskivirhe test datassa on %.f" % mean_absolute_error(df_validation['CloseFuture'], df_validation['Ennuste']))
```

## 2 Tehtävä 2

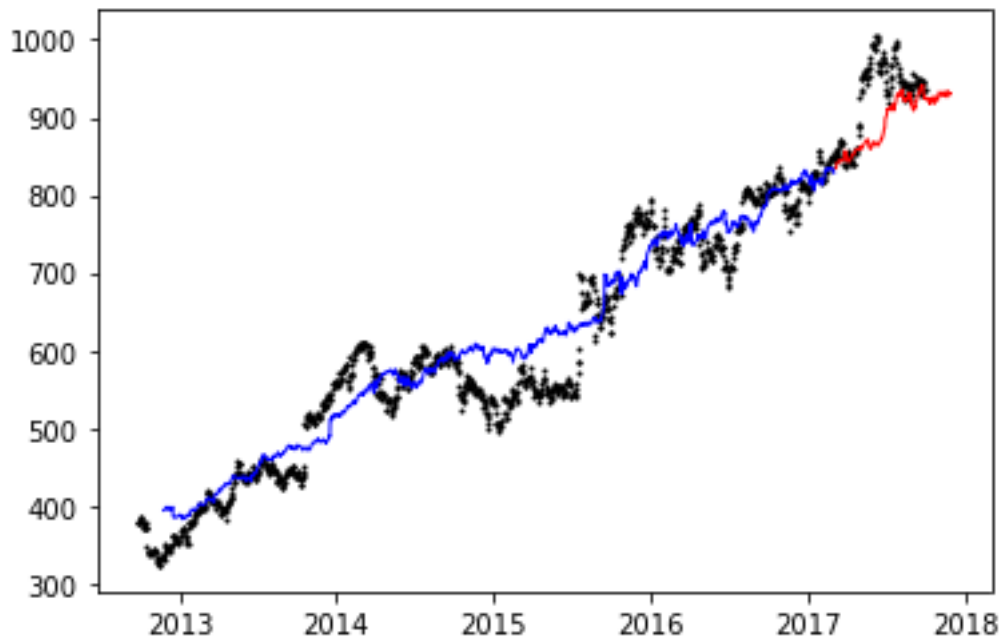
a) 7 päivän ennuste tulevaisuuteen



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')
```

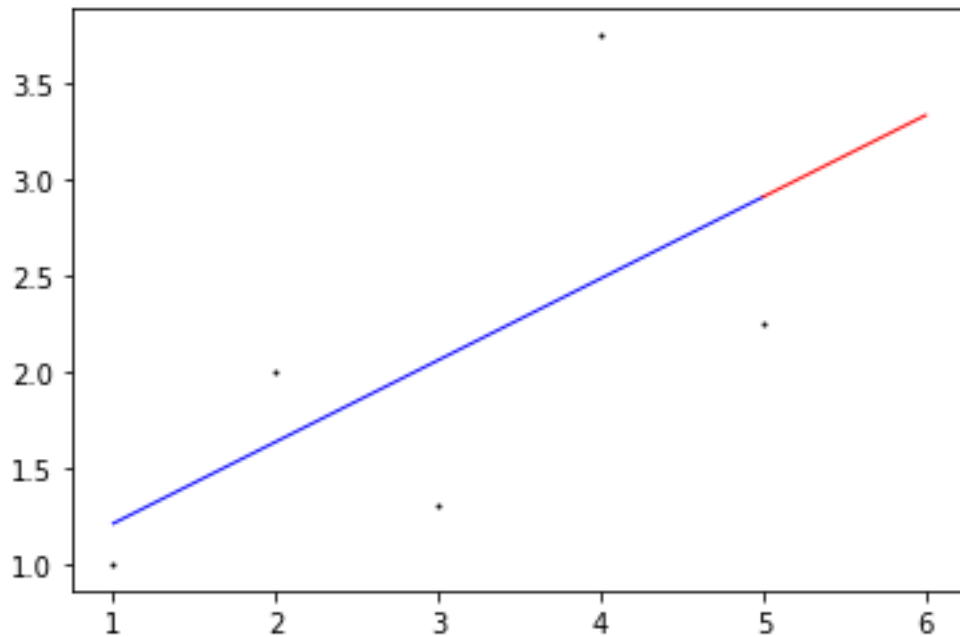
```
plt.show()
```

```
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)

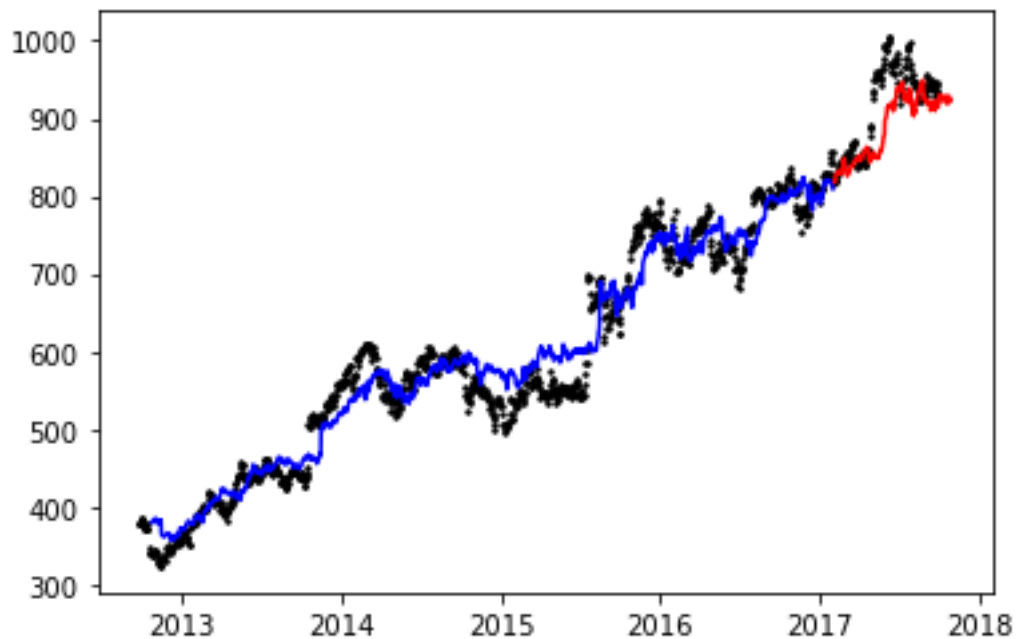
X_test = np.array(df_test['X'])
X_test = X_test.reshape(-1,1) # vain jos yksiulotteinen taulukko
df_test['Ennuste'] = model.predict(X_test)

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'])

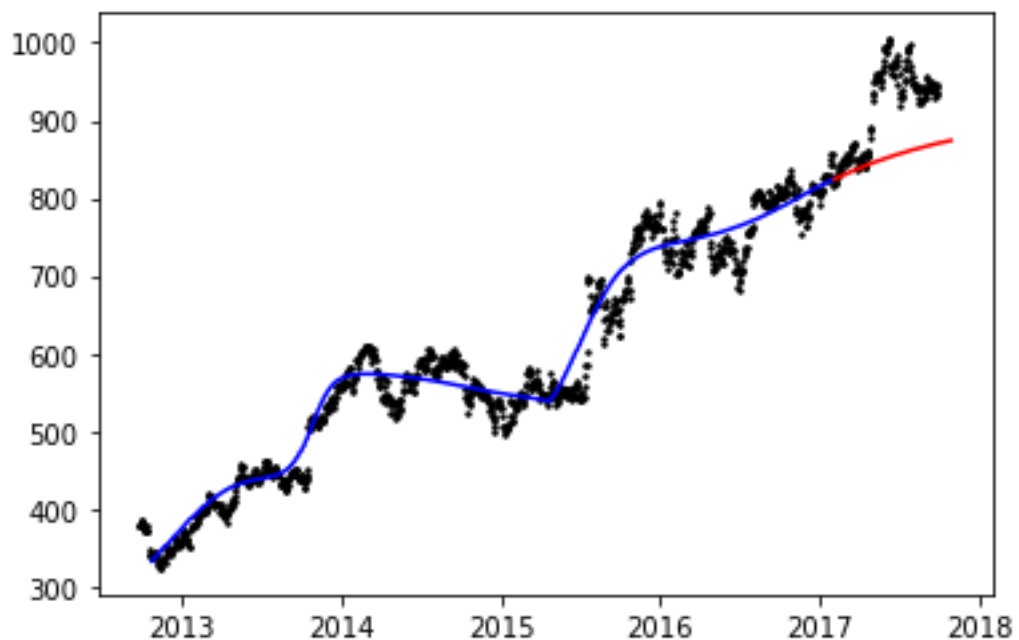
# epochs = kuinka monta kertaa opetusdata käydään läpi training vaiheessa
# (painotus), batch_size = kuinka monen data rivin jälkeen painokertoimia päivitetään
# (oppiminen)
model.fit(X_scaled, y, epochs = 100, batch_size = 10)
ennuste_train = model.predict(X_scaled)
df_train['Ennuste'] = ennuste_train

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
```



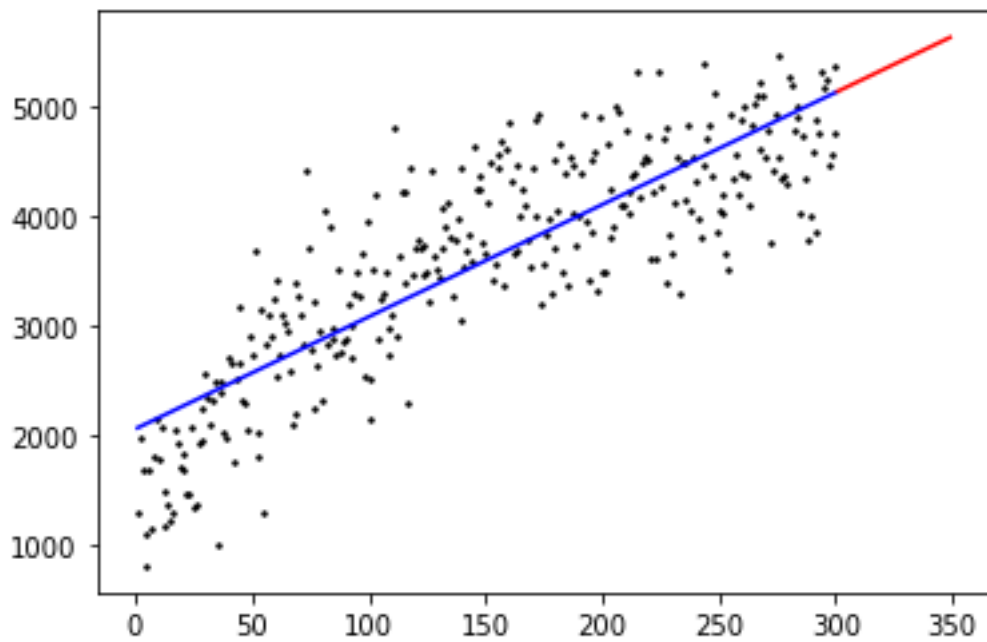
```
X_test = np.array(df_test[['Time']])
X_test = X_test.reshape(-1,1)
X_testscaled = scaler.transform(X_test)
ennuste_test = model.predict(X_testscaled)
df_test['Ennuste'] = ennuste_test

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)
```



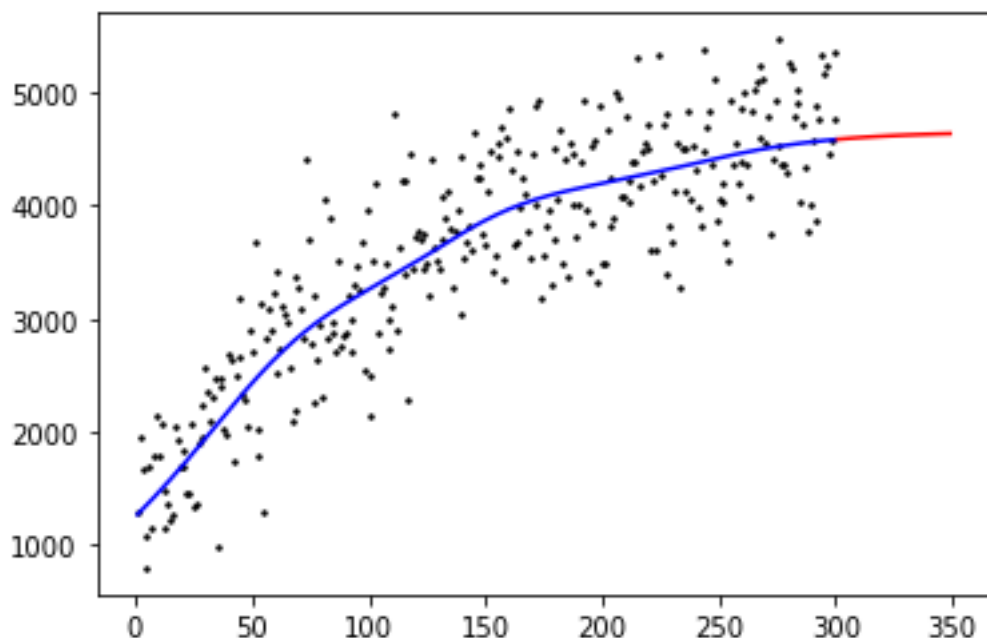
```
ennuste_train = model.predict(X)
df_train['Ennuste'] = ennuste_train
```

```
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_scon	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_score	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']))
```



```
# 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. piilotettu 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	assengerl	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-Lähimmänaapurin 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	PassengerId	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. piilotettu 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)
```

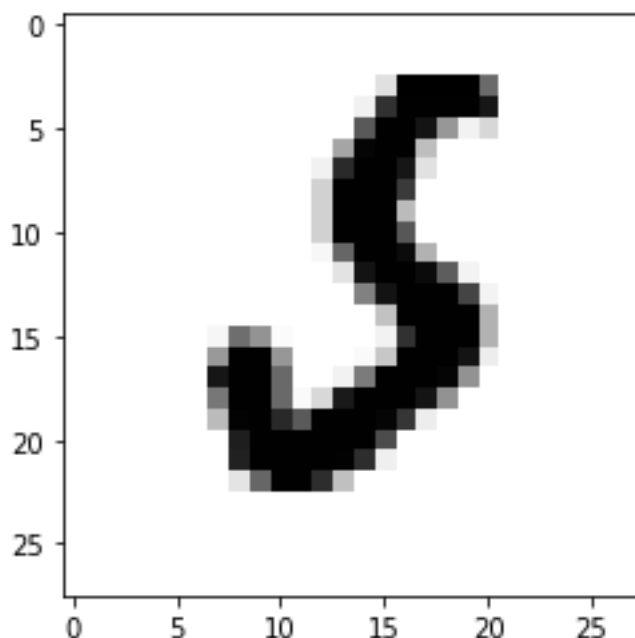
```
# test
X = np.array(df_test[input_variables])
#y = np.array(pd.get_dummies(df_test['Survived']))
scaler = preprocessing.StandardScaler()
X_scaled = scaler.fit_transform(X)

# hakee sarakkeesta ennusteen, jonka todennäköisyys suurin
ennuste = np.argmax(model.predict(X_scaled), axis=1)
df_test['Ennuste'] = ennuste

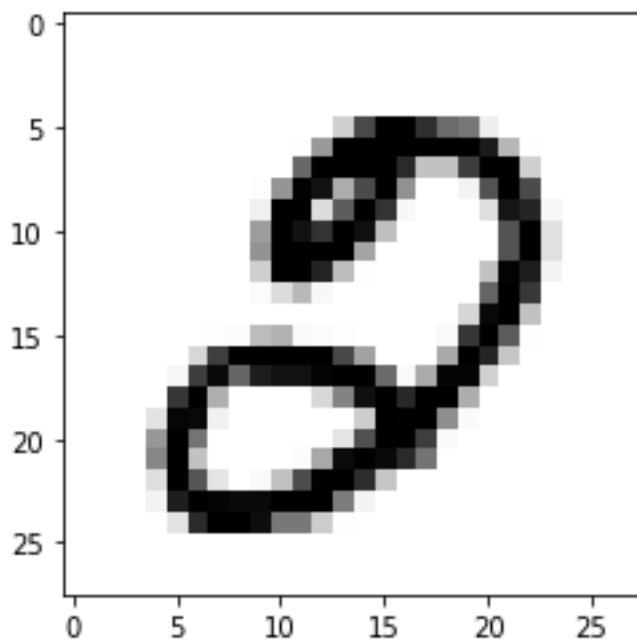
results_fields = ['PassengerId', 'Survived', 'Ennuste']
df_results = df_test[results_fields].sample(20)
```

## 11 Tehtävä 11

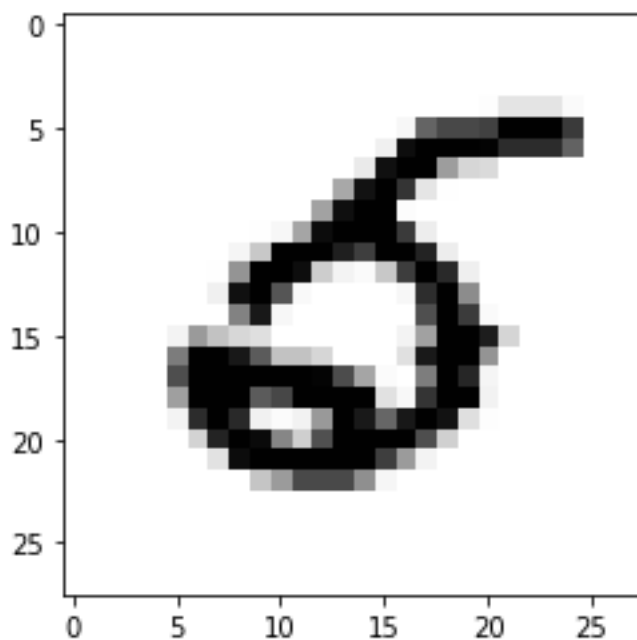
Kategorointitarkkuus opetusdatassa saavutti 99.5% tarkkuuden ja testidatassa 98.0 % tarkkuuden.



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')
])

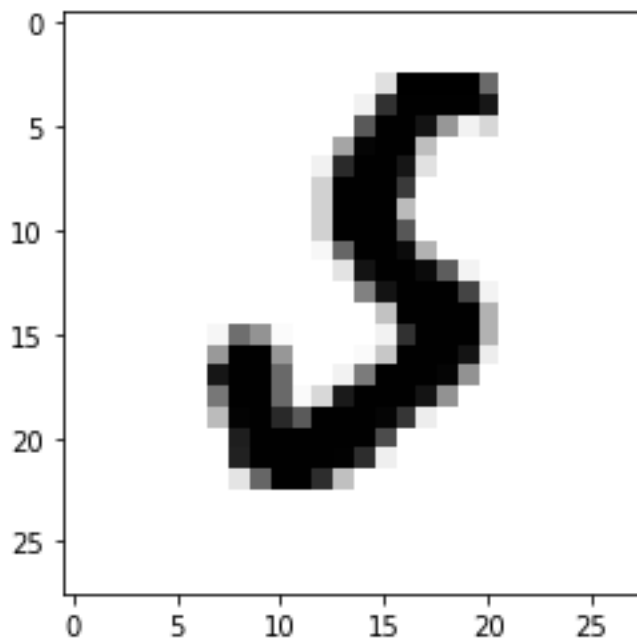
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics=['categorical_accuracy'])

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.



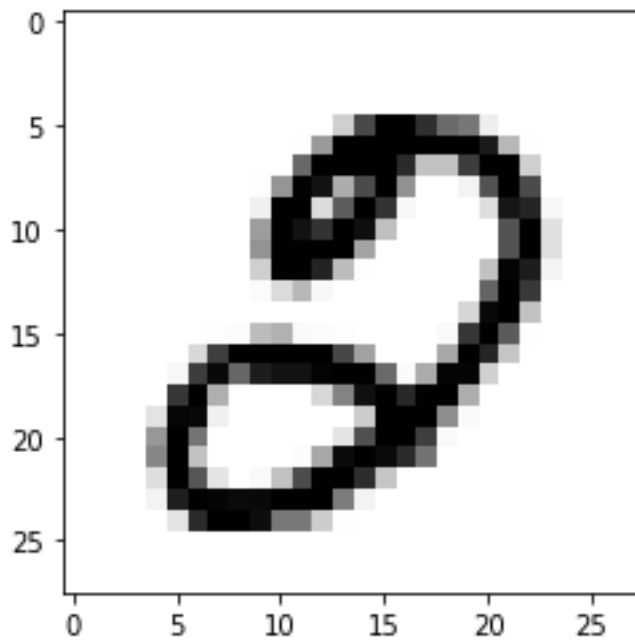
Käytetty malli tunnistaa yllä olevan kuvan 98 % todennäköisyydellä numeroksi 5.

Edellinen malli tunnistaa yllä olevan kuvan 37 % todennäköisyydellä numeroksi 3, 31 % todennäköisyydellä numeroksi 5 ja 30 % todennäköisyydellä numeroksi 6.

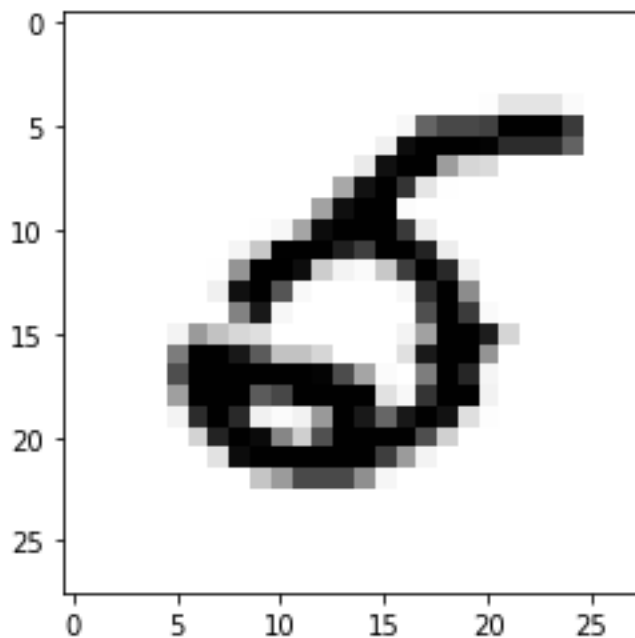
Ihmissilmä näkee tuossa numeron 5.



# jamk.fi



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')
```

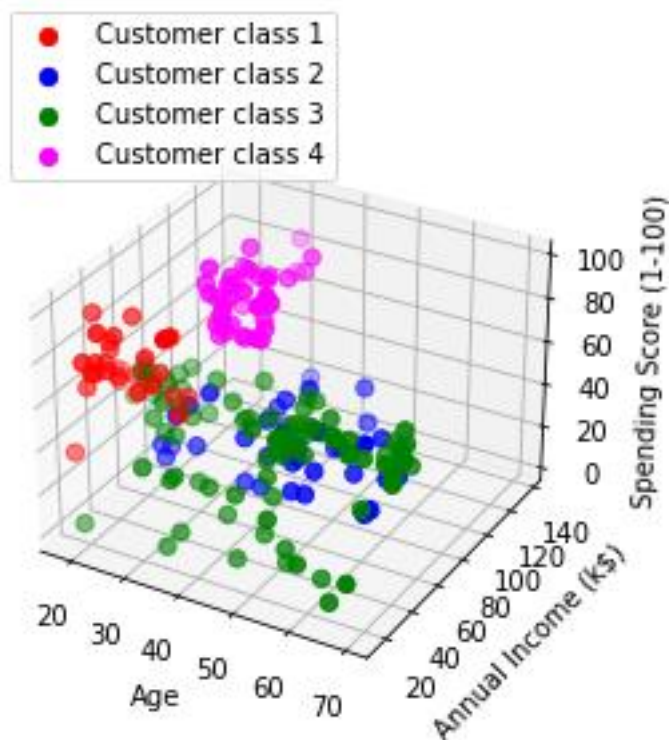
## 13 Tehtävä 13

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)

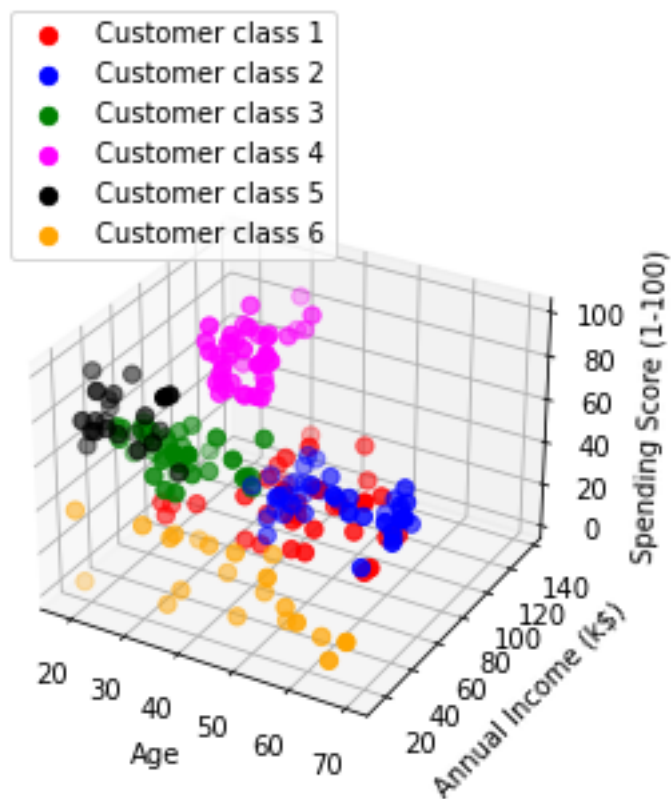
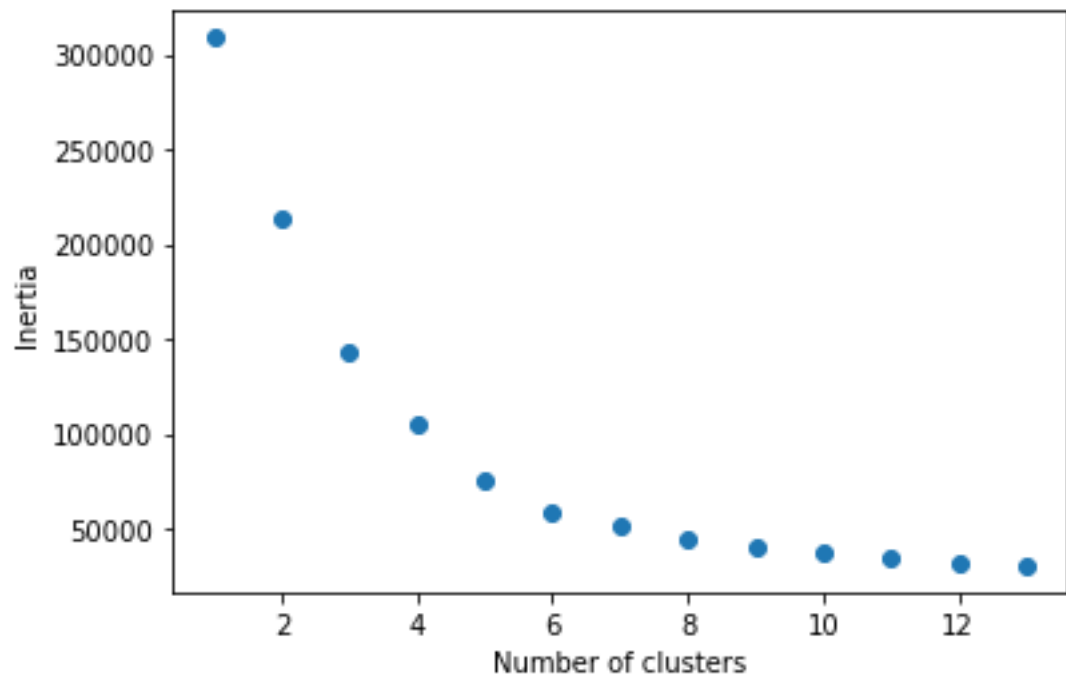
labels = model.labels_
df['Label'] = labels

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()
```

## 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()

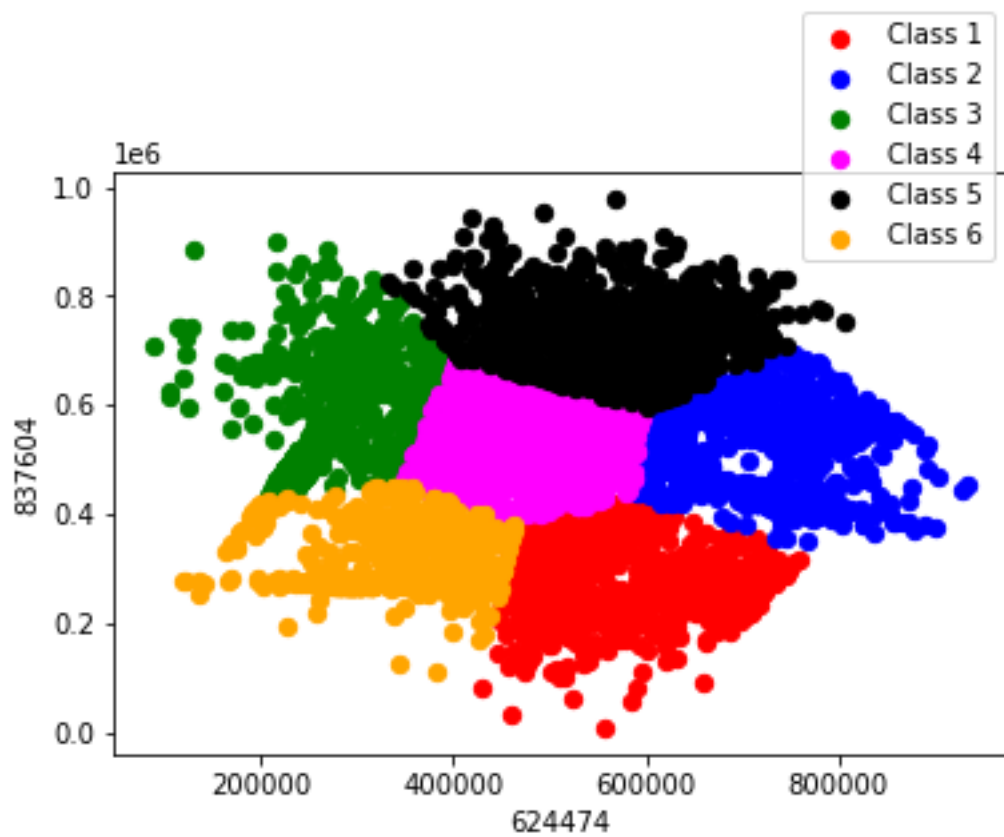
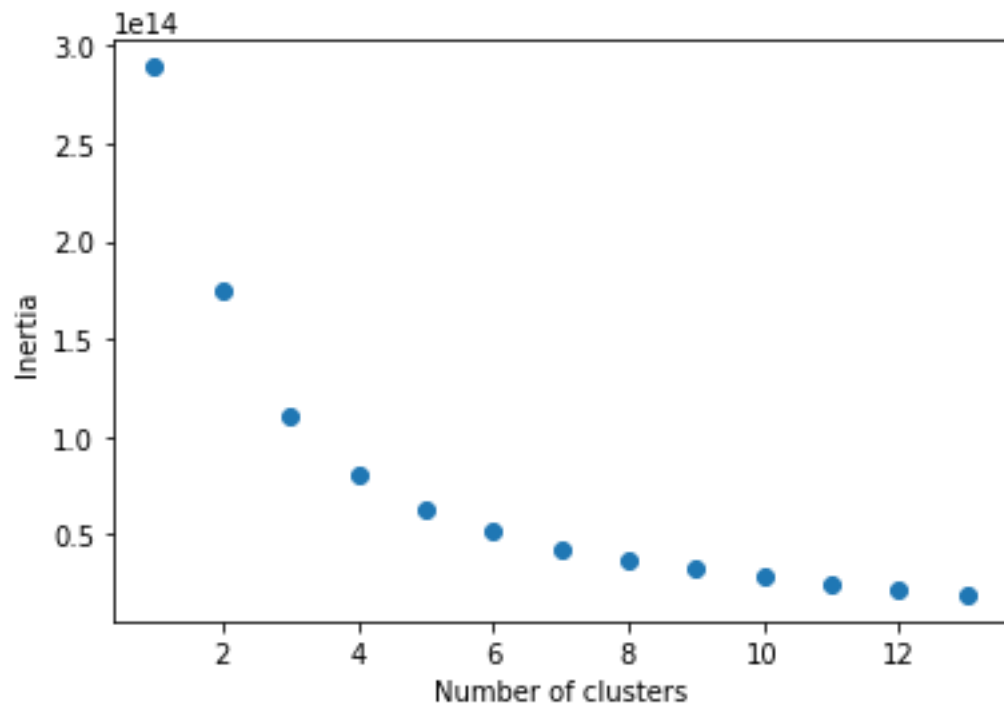
model = KMeans(n_clusters=6)
model.fit(X)
labels = model.labels_
df['Label'] = labels

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()
```

## 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

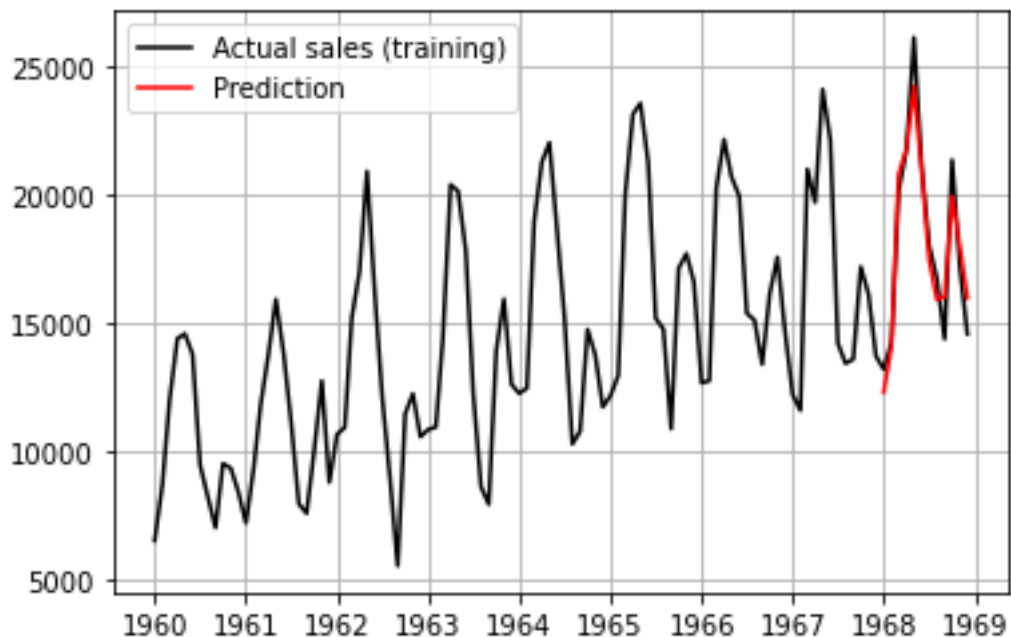
colors = {0:'red', 1:'blue', 2:'green', 3:'magenta', 4:'black', 5:'orange'}

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()
```



## 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)

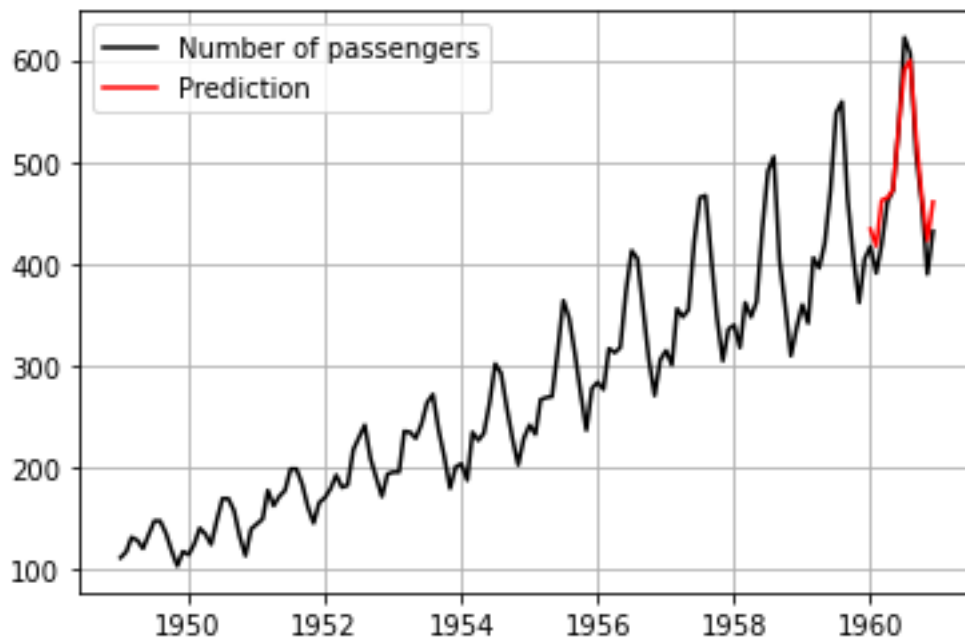
# Trendin mallinnus lineaarisella regressiolla
modellLR = linear_model.LinearRegression()
XLR = df_train['Time'].values
XLR = XLR.reshape(-1,1)
yLR = df_train['Sales'].values
yLR = yLR.reshape(-1,1)
modellLR.fit(XLR, yLR)
XLR_test = df_test['Time'].values
XLR_test = XLR_test.reshape(-1,1)
df_test['SalesAvgPred'] = modellLR.predict(XLR_test)

# trendin kulmakerroin
slope = modellLR.coef_

# LSTM verkon muodostus ja opetus
modellLSTM = tf.keras.Sequential([
```



## 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
modellLR = linear_model.LinearRegression()
XLR = df_train['Time'].values
XLR = XLR.reshape(-1,1)
yLR = df_train['Passengers'].values
yLR = yLR.reshape(-1,1)
modellLR.fit(XLR, yLR)
XLR_test = df_test['Time'].values
XLR_test = XLR_test.reshape(-1,1)
df_test['PassengersAvgPred'] = modellLR.predict(XLR_test)

# trendin kulmakerroin
slope = modellLR.coef_
```



```
# LSTM verkon muodostus ja opetus
modellLSTM = 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)
])

modellLSTM.compile(
    loss='mse',
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    metrics=['mae'])

modellLSTM.fit(X_scaledLSTM, y_scaled, epochs=200, batch_size=seqlength)

# Ennusteen (ennusteDiff + trendi) määrittäminen
ennusteDiff = scaler.inverse_transform(
    modellLSTM.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))
```

## 19 Tehtävä 19

Sd

### 19.1 Lähdekoodi

sds

## 20 Tehtävä 20: Lähteiden asiakkaiden tunnistaminen

Oheisessa datassa (Telco.csv) on teleoperaattorin asiakastietokanta. Muuttuja churn=1 ilmaisee, että asiakas on lopettanut sopimuksen. Tehtäväsi on ennustaa kunkin asiakkaan "churn"-kentän arvo käyttämällä input-muuttujina muiden kenttien arvoja. Jätä datasta 100 satunnaista asiakasta test-dataksi, ja vssalitse loput training-dataksi, jonka avulla muodostat haluamasi koneoppimismallin. Raportoi vastaukseesi seuraavat asiat:

- Käyttämäsi koneoppimismalli ja sen tarkkuus training- ja test-datassa.
- Huolehdi siitä, ettei malli ole liikaa ylisovitettu.
- Listaus 20 satunnaisesta test-datan asiakkaasta, jossa näkyy kunkin asiakkaan todellinen churn-arvo sekä mallisi ennuste (churn-riski numerona välillä 0-1)

Sd

### 20.1 Lähdekoodi

sds