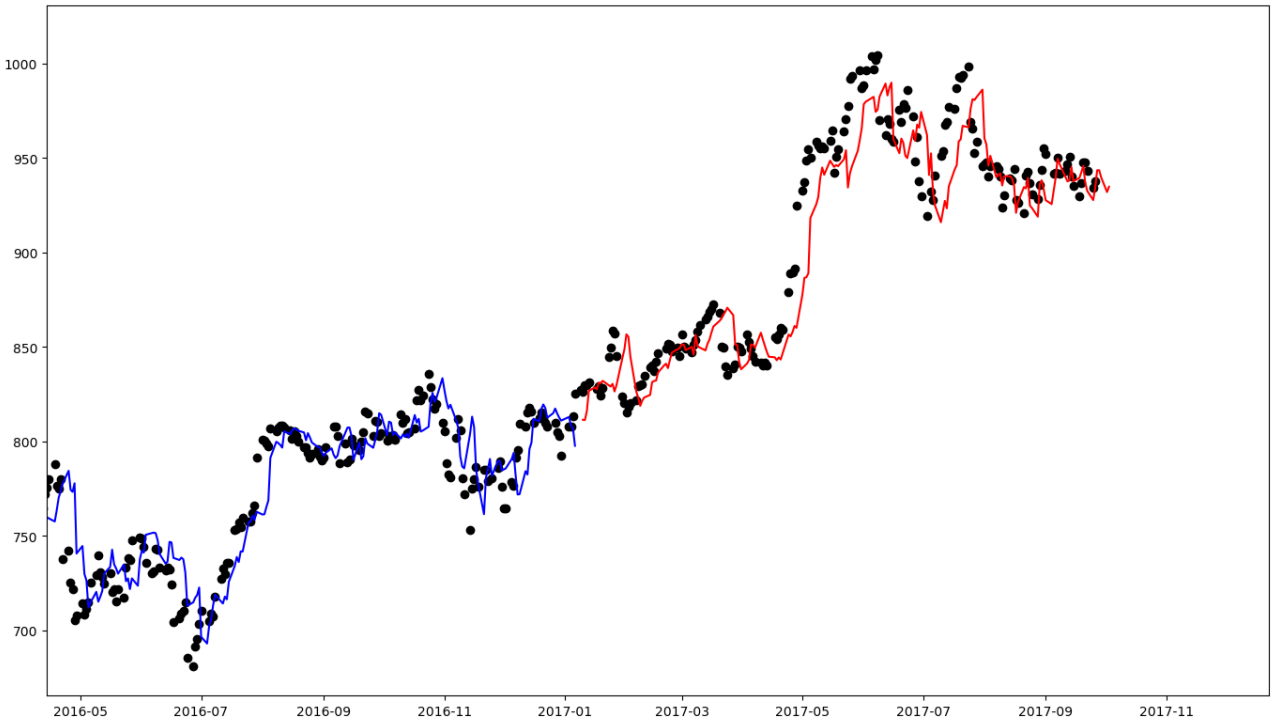
Tehtävä 1. a

Ennusteen keskivirhe test datassa on 70

Tehtävä 1. b

Ennusteen keskivirhe test datassa on 35

Tehtävä 2. a



Ennusteen keskivirhe test datassa on 20

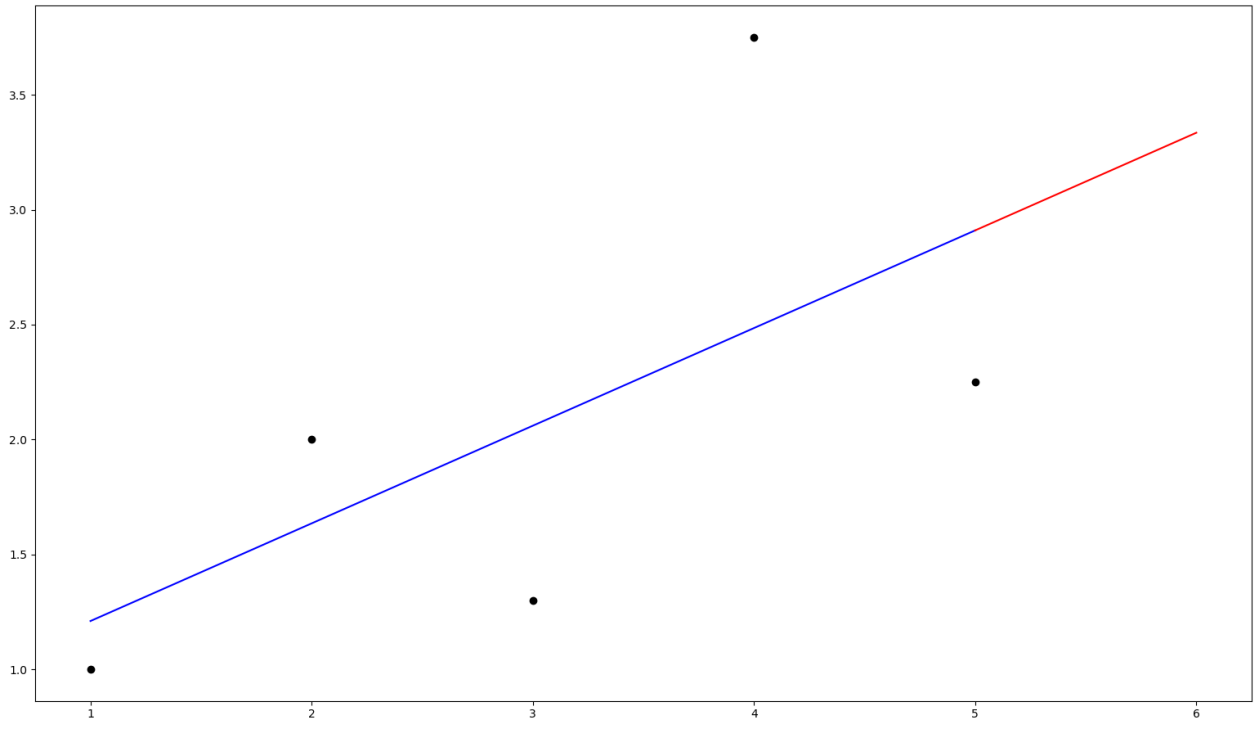
Ennusteen keskivirhe training datassa on 16

Tehtävä 2. b

Ennusteen keskivirhe test datassa on 60

Ennusteen keskivirhe training datassa on 34

Tehtävä 3.



Ennuste muuttujan y arvoksi x:n arvolla 6 saadaan 3.335

Tehtävä 4.

Ennusteen keskivirhe test datassa on 35

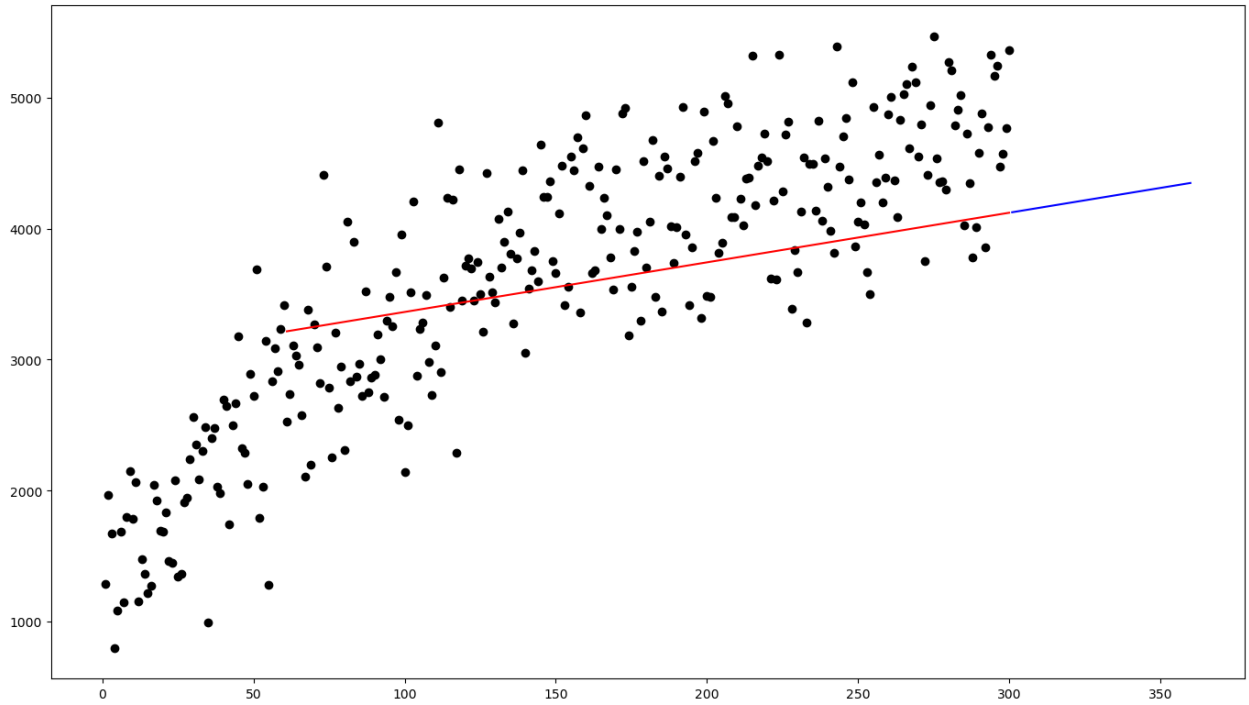
mae: 30.2372 (training datan keskivirhe)

Tehtävä 5.

Ennusteen keskivirhe test datassa on 95

Mean absolute error: 19.1957

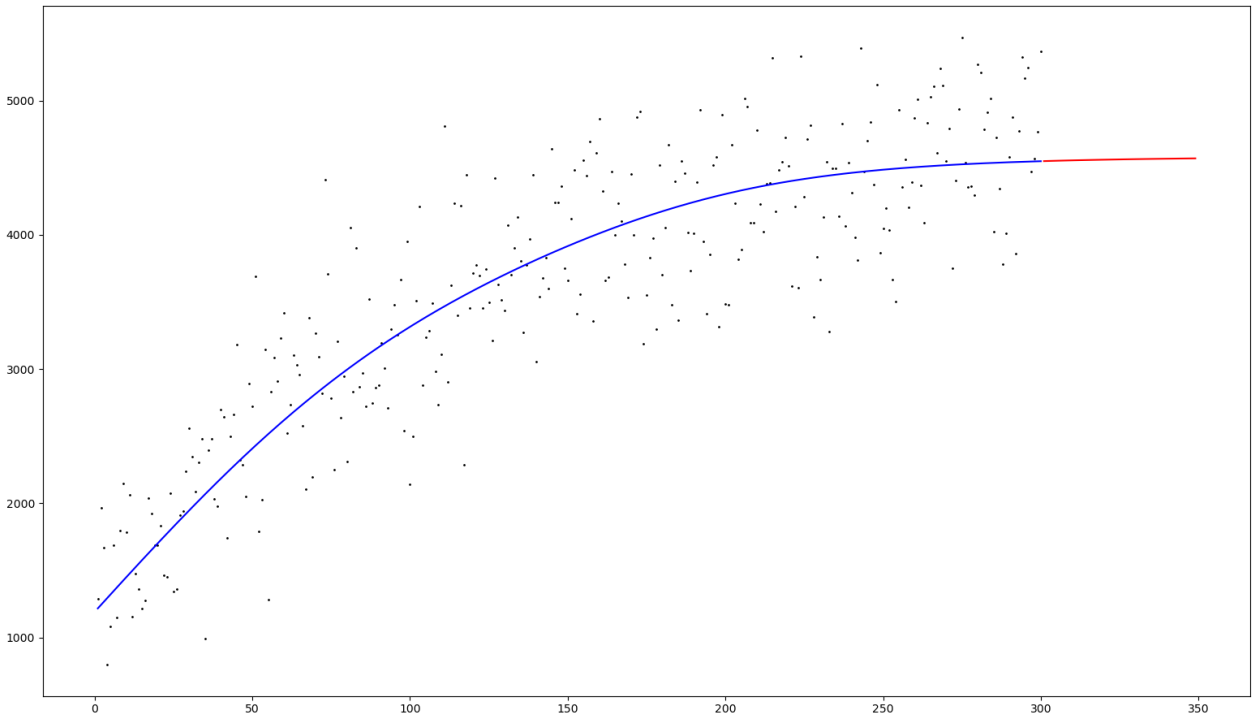
Tehtävä 6a.



Ennusteen keskivirhe test datassa on 880

Ennusteen keskivirhe training datassa on 410

Tehtävä 6b.

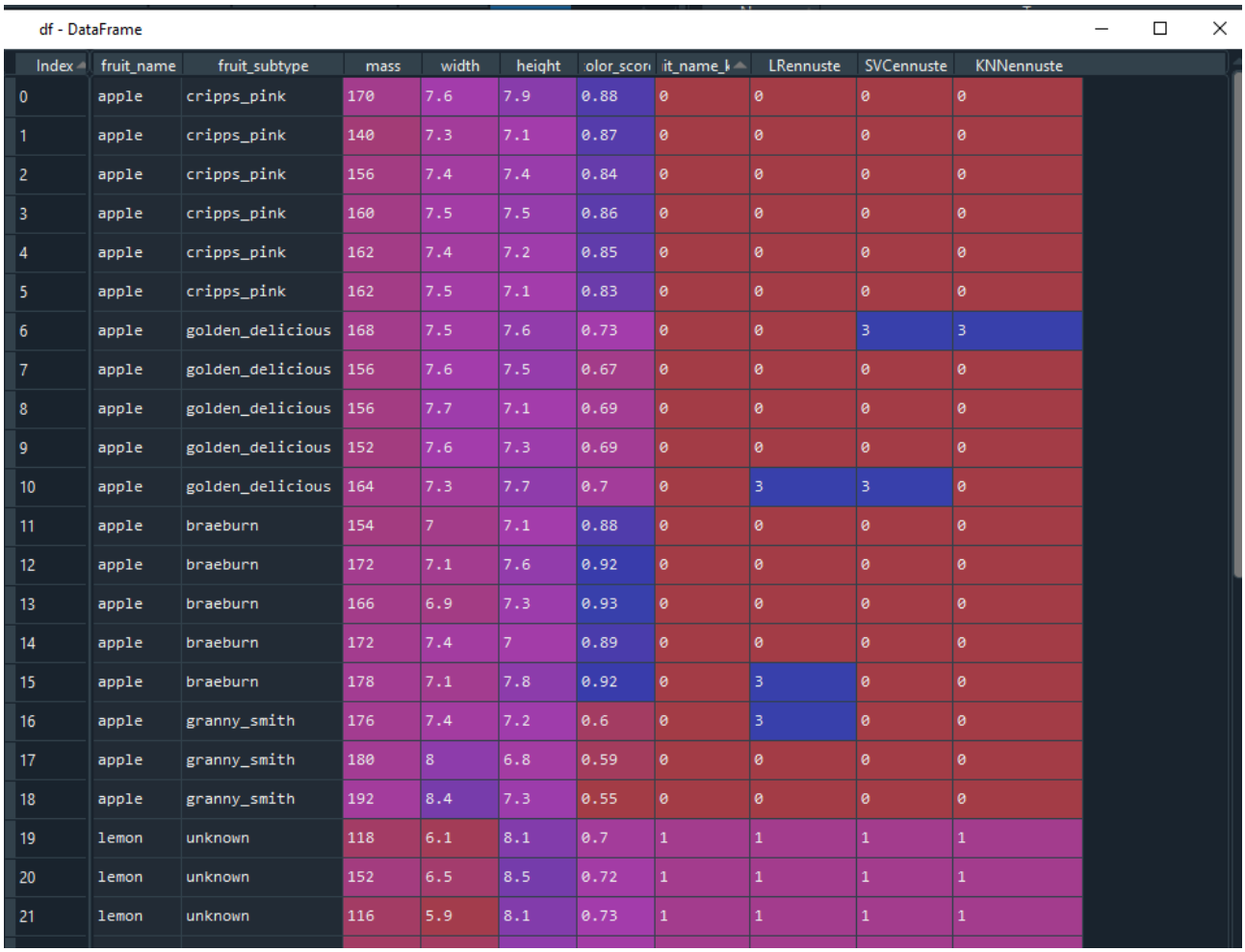


Ennusteen keskivirhe training datassa on 408

mae: 406.0471

MLP neuroverkolla tehty malli on mielestäni tarkempi.

Tehtävä 7.



df - DataFrame

Index	fruit_name	fruit_subtype	mass	width	height	olor_scor	it_name_k	LRennuste	SVCennuste	KNNennuste
0	apple	cripps_pink	170	7.6	7.9	0.88	0	0	0	0
1	apple	cripps_pink	140	7.3	7.1	0.87	0	0	0	0
2	apple	cripps_pink	156	7.4	7.4	0.84	0	0	0	0
3	apple	cripps_pink	160	7.5	7.5	0.86	0	0	0	0
4	apple	cripps_pink	162	7.4	7.2	0.85	0	0	0	0
5	apple	cripps_pink	162	7.5	7.1	0.83	0	0	0	0
6	apple	golden_delicious	168	7.5	7.6	0.73	0	0	3	3
7	apple	golden_delicious	156	7.6	7.5	0.67	0	0	0	0
8	apple	golden_delicious	156	7.7	7.1	0.69	0	0	0	0
9	apple	golden_delicious	152	7.6	7.3	0.69	0	0	0	0
10	apple	golden_delicious	164	7.3	7.7	0.7	0	3	3	0
11	apple	braeburn	154	7	7.1	0.88	0	0	0	0
12	apple	braeburn	172	7.1	7.6	0.92	0	0	0	0
13	apple	braeburn	166	6.9	7.3	0.93	0	0	0	0
14	apple	braeburn	172	7.4	7	0.89	0	0	0	0
15	apple	braeburn	178	7.1	7.8	0.92	0	3	0	0
16	apple	granny_smith	176	7.4	7.2	0.6	0	3	0	0
17	apple	granny_smith	180	8	6.8	0.59	0	0	0	0
18	apple	granny_smith	192	8.4	7.3	0.55	0	0	0	0
19	lemon	unknown	118	6.1	8.1	0.7	1	1	1	1
20	lemon	unknown	152	6.5	8.5	0.72	1	1	1	1
21	lemon	unknown	116	5.9	8.1	0.73	1	1	1	1

Logistisen regressiomallin tarkkuus: 0.8813559322033898

SVM mallin tarkkuus: 0.9661016949152542

KNN mallin tarkkuus: 0.9830508474576272

Tehtävä 8.

df - DataFrame

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
16	apple	golden_delicious	156	7.6	7.5	0.67	0
17	apple	golden_delicious	168	7.5	7.6	0.73	0

Format Resize Background color Column min/max

Mallin tarkkuus training datanasta: categorical_accuracy: 1.0000 – 100%

Tehtävä 9.

df_otos - DataFrame

Index	PassengerId	Survived	KNNEnnuste
1	2	1	1
528	529	0	0
774	775	1	1
51	52	0	0
520	521	1	1
107	108	1	0
26	27	0	0
427	428	1	1
662	663	0	0
511	512	0	0
606	607	0	0
347	348	1	0
375	376	1	1
694	695	0	0
862	863	1	1
583	584	0	1
109	110	1	1
633	634	0	0
810	811	0	0
678	679	0	0

Logistisen regressiomallin tarkkuus (training data): 78.87 %

Logistisen regressiomallin tarkkuus test datassa: 79.0 %

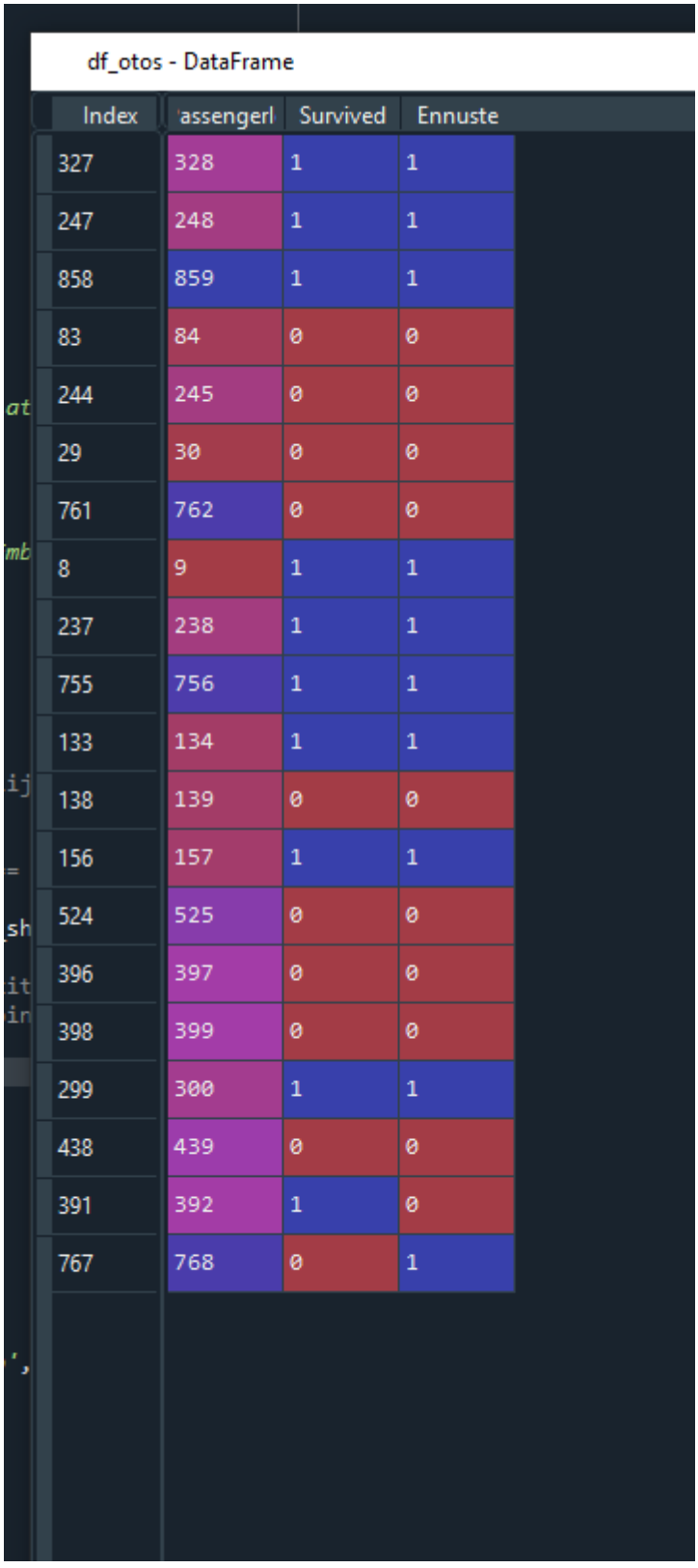
SVM mallin tarkkuus (training data): 82.92 %

SVM tarkkuus test datassa: 84.0 %

KNN mallin tarkkuus (training data): 83.94 %

KNN tarkkuus test datassa: 82.0 %

Tehtävä 10.



df_otos - DataFrame

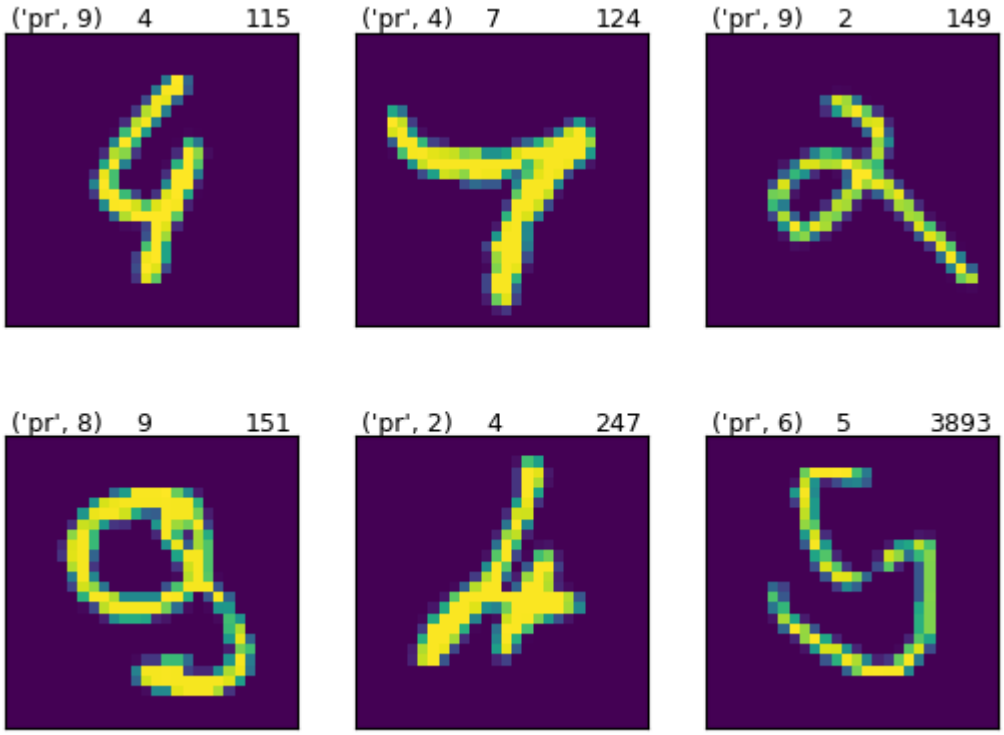
Index	assengerI	Survived	Ennuste
327	328	1	1
247	248	1	1
858	859	1	1
83	84	0	0
244	245	0	0
29	30	0	0
761	762	0	0
8	9	1	1
237	238	1	1
755	756	1	1
133	134	1	1
138	139	0	0
156	157	1	1
524	525	0	0
396	397	0	0
398	399	0	0
299	300	1	1
438	439	0	0
391	392	1	0
767	768	0	1

Tarkkuus training datassa: 82.92% (0.8292)

tarkkuus test datassa: 81.5 %

Tehtävä 11.

Ensiksi kuvan yläpuolella on mallin ennustama numero (Pr) ja perässä on oikea numero. Eli nämä ovat menneet väärin.

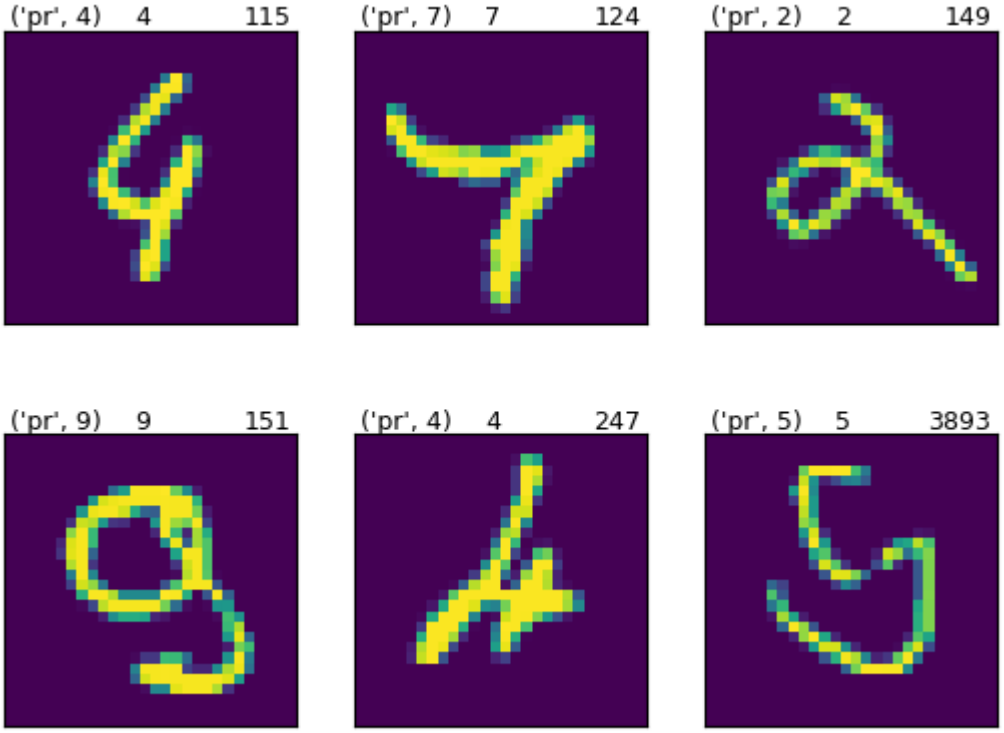


Kategorinen tarkkuus /training: 99,5% -> virhe 0,5%

Kategorinen tarkkuus /test: 98,3% -> virhe 1,7%

Tehtävä 12.

Ensiksi kuvan yläpuolella on mallin ennustama numero (Pr) ja perässä on oikea numero. Eli nämä ovat menneet nyt kaikki oikein.



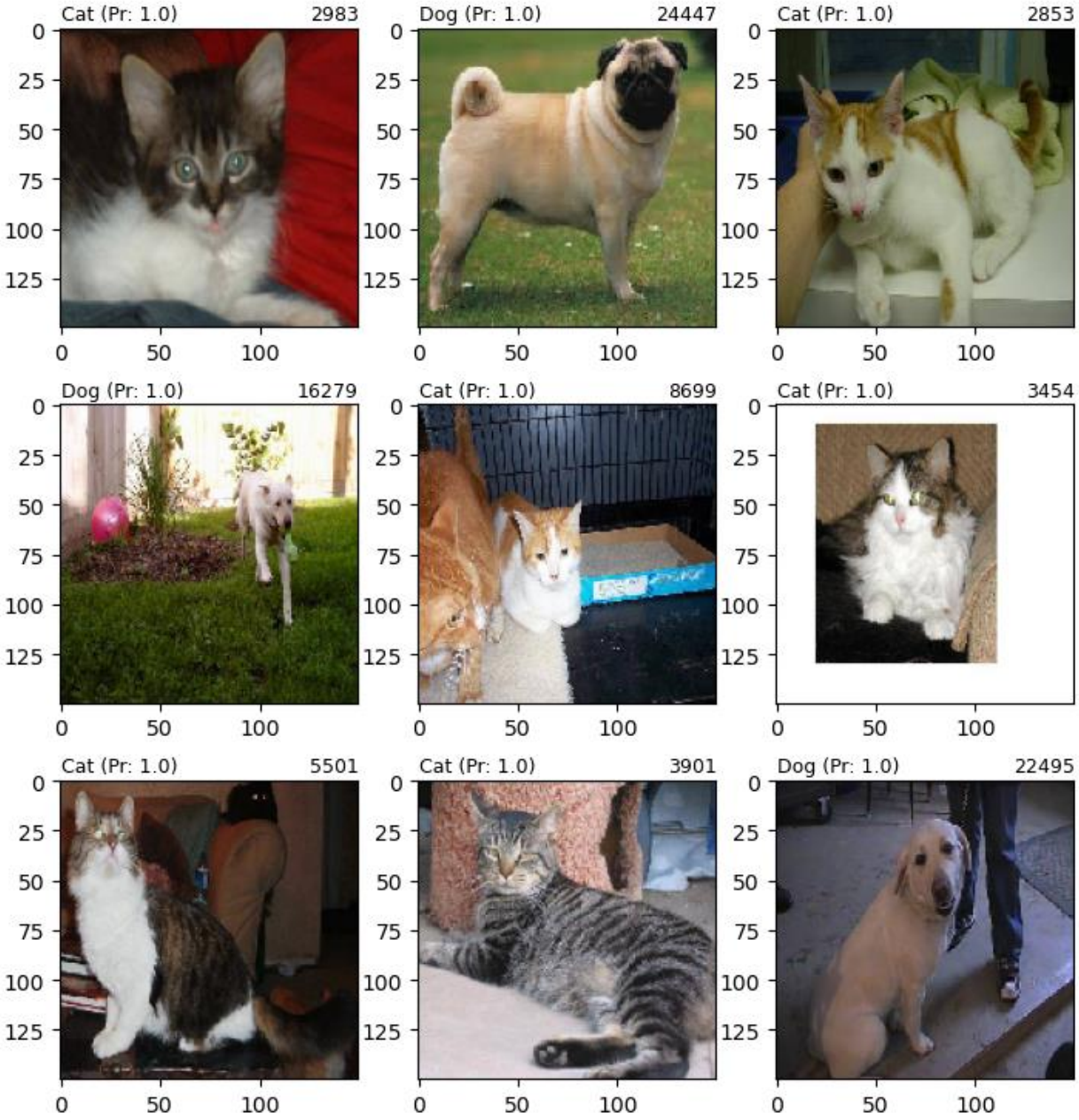
Kategorinen tarkkuus /training: 99,6% -> virhe 0,4%

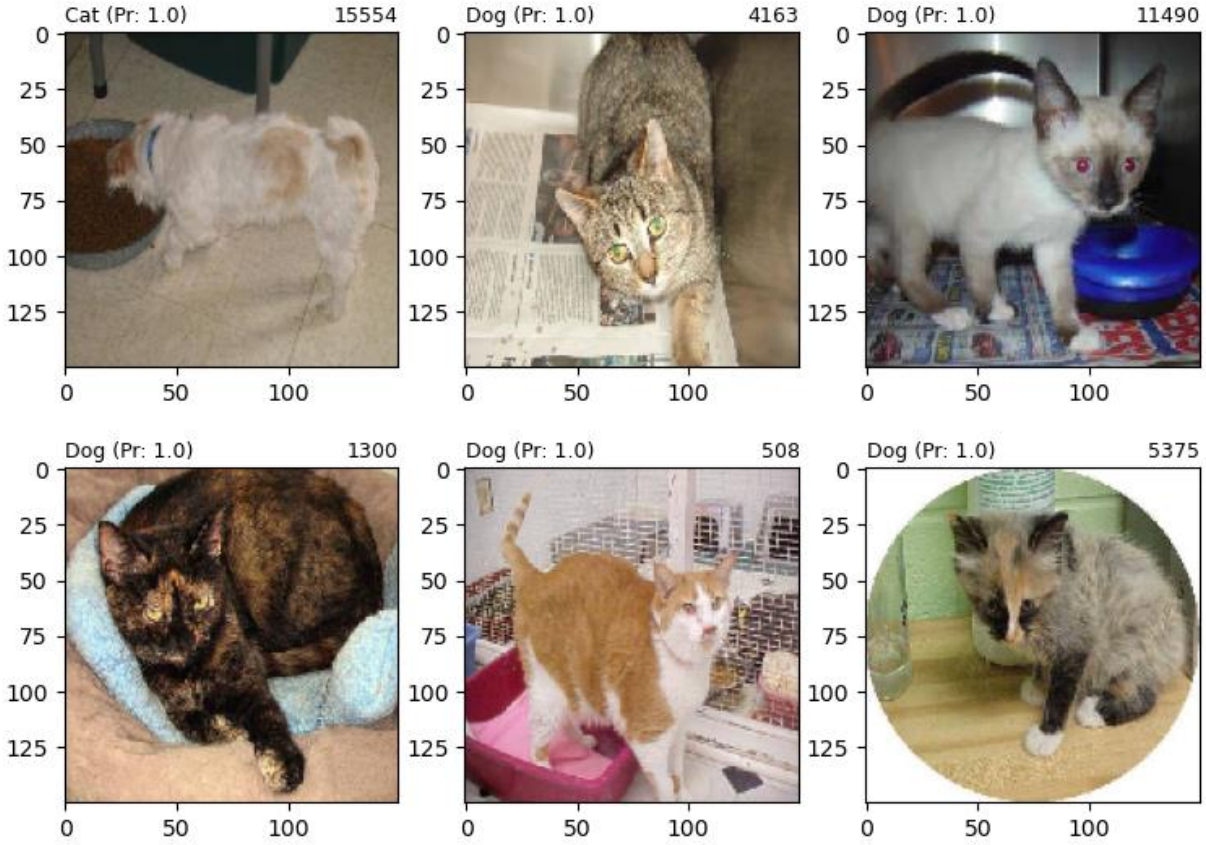
Kategorinen tarkkuus /test: 99,0% -> virhe 1%

Tehtävä 13.

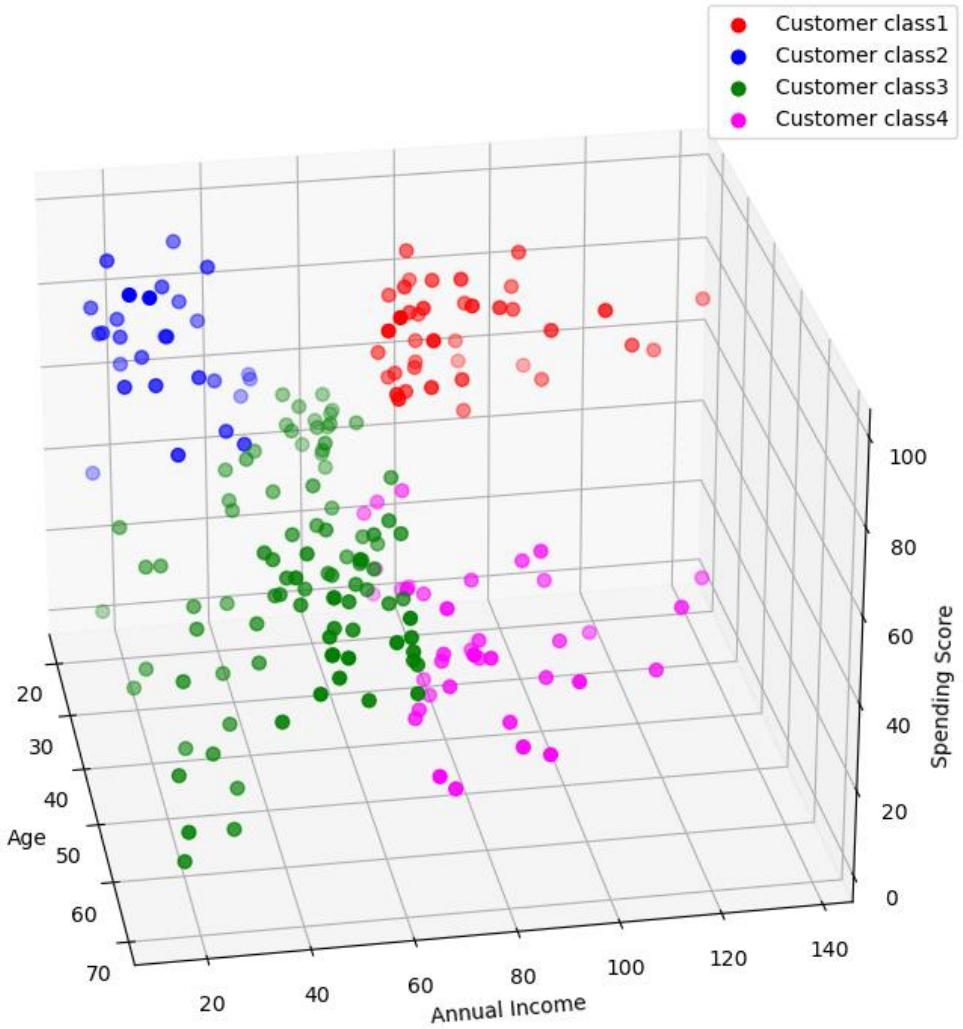
Validoitu tarkkuus 0,8170

acc: 0.9289 - val_acc: 0.8170

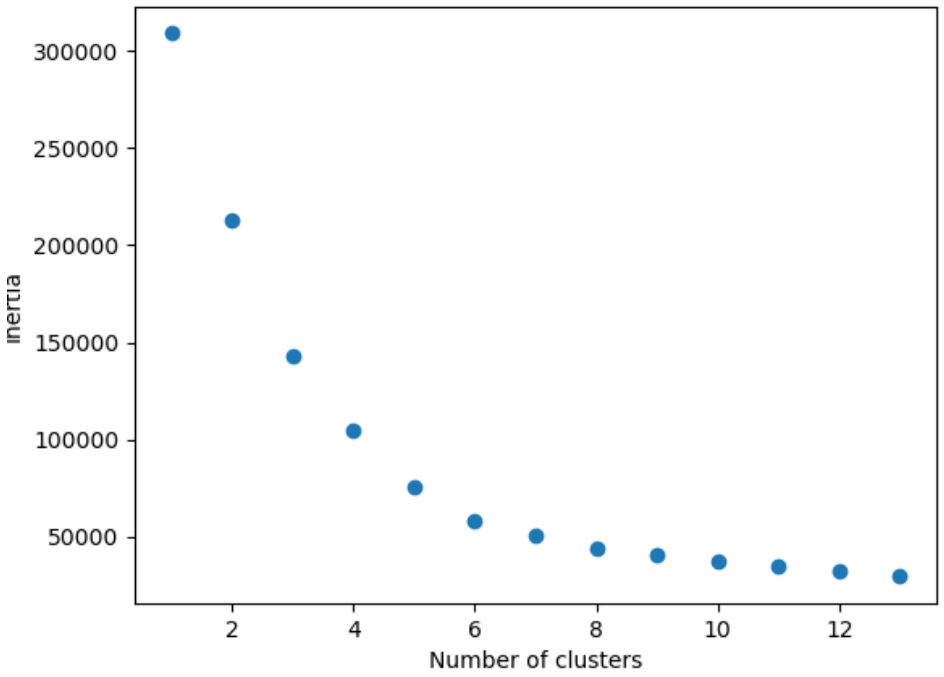


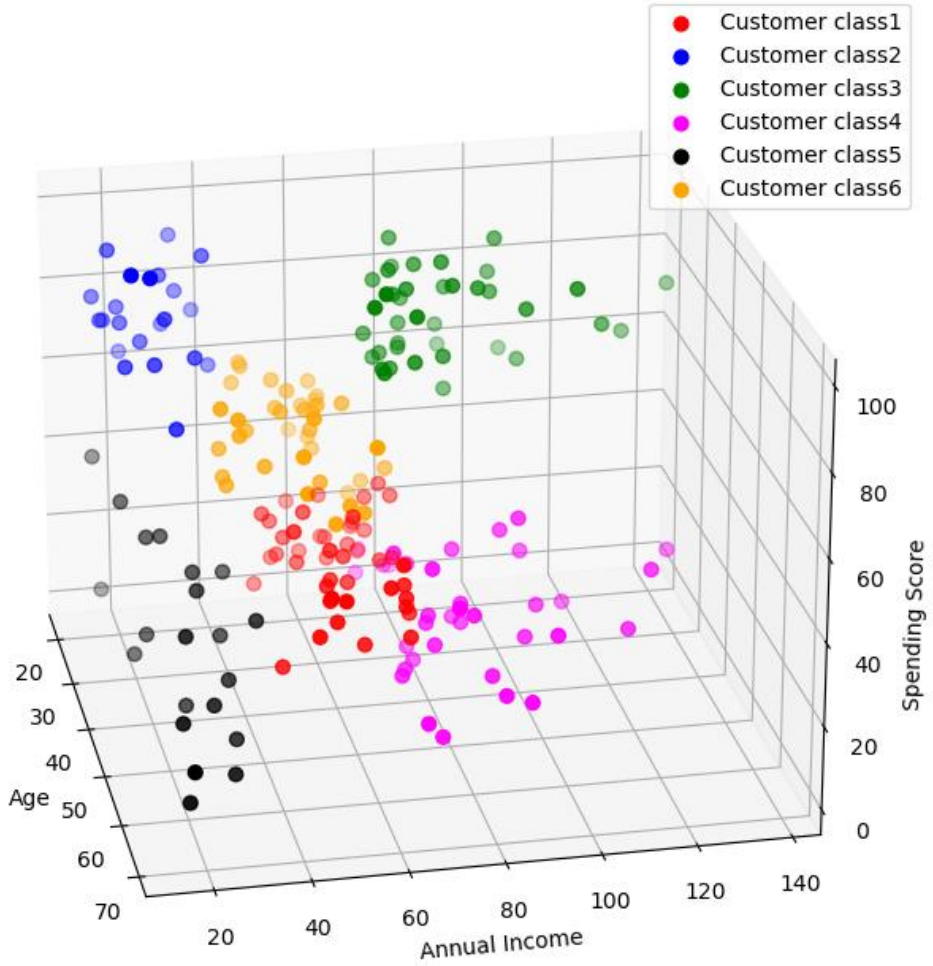

Cat (Pr: 1.0) 15554
Dog (Pr: 1.0) 4163
Dog (Pr: 1.0) 11490
Dog (Pr: 1.0) 1300
Dog (Pr: 1.0) 508
Dog (Pr: 1.0) 5375

Tehtävä 14.

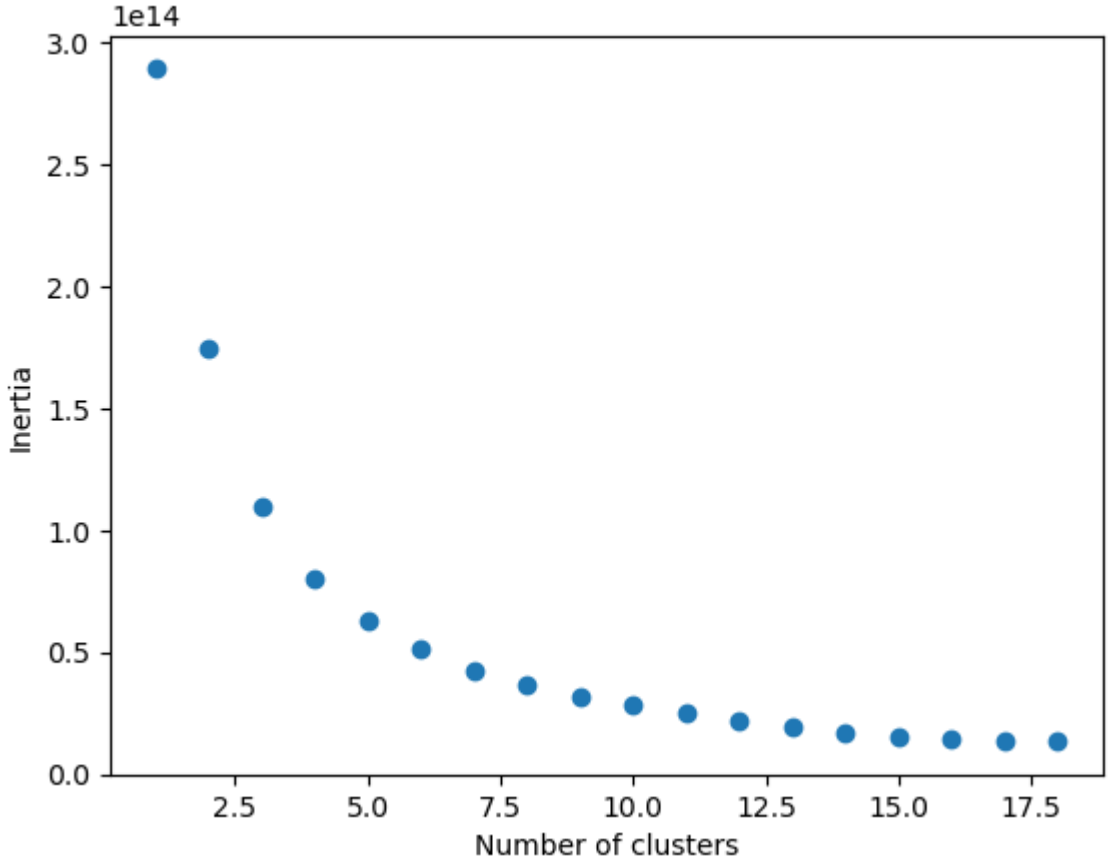

Customer class1
Customer class2
Customer class3
Customer class4
Age
20
30
40
50
60
70
Annual Income
20
40
60
80
100
120
140
Spending Score
0
20
40
60
80
100

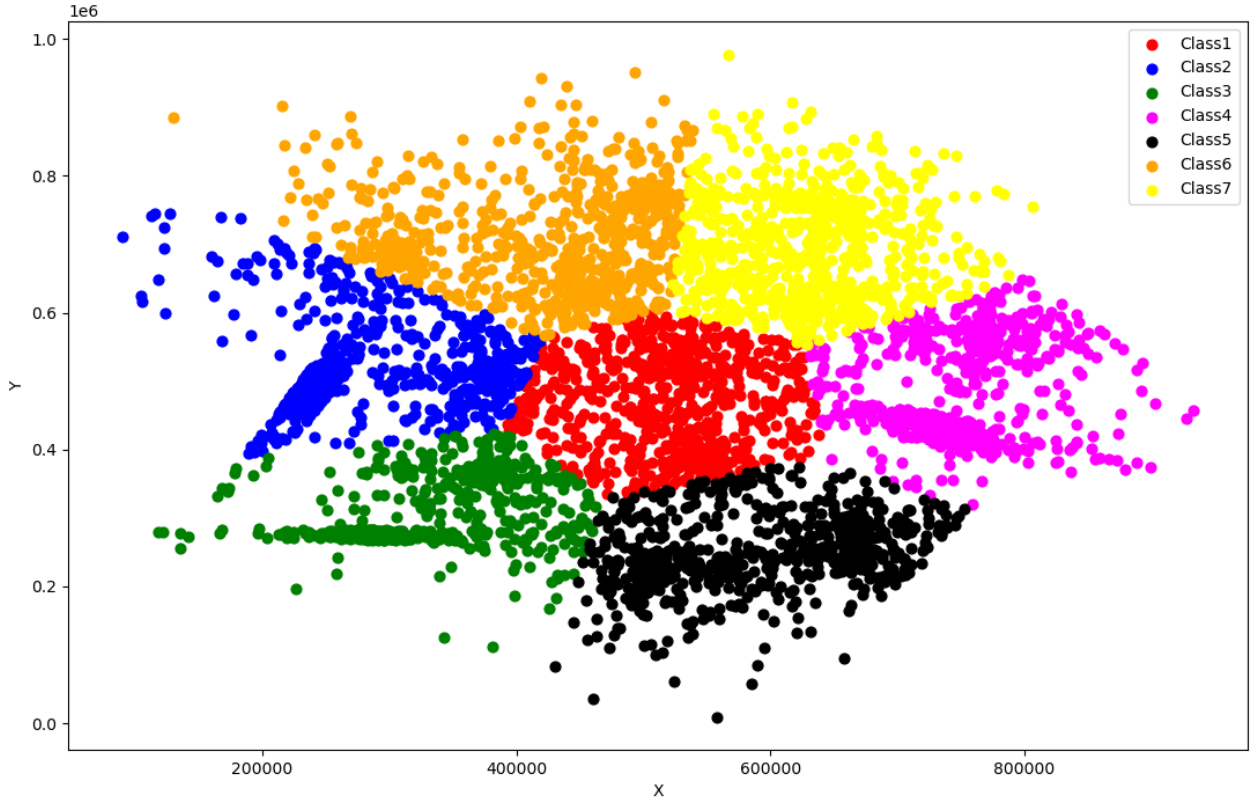
Tehtävä 15.


300000
250000
200000
150000
100000
50000
Inertia
2
4
6
8
10
12
Number of clusters

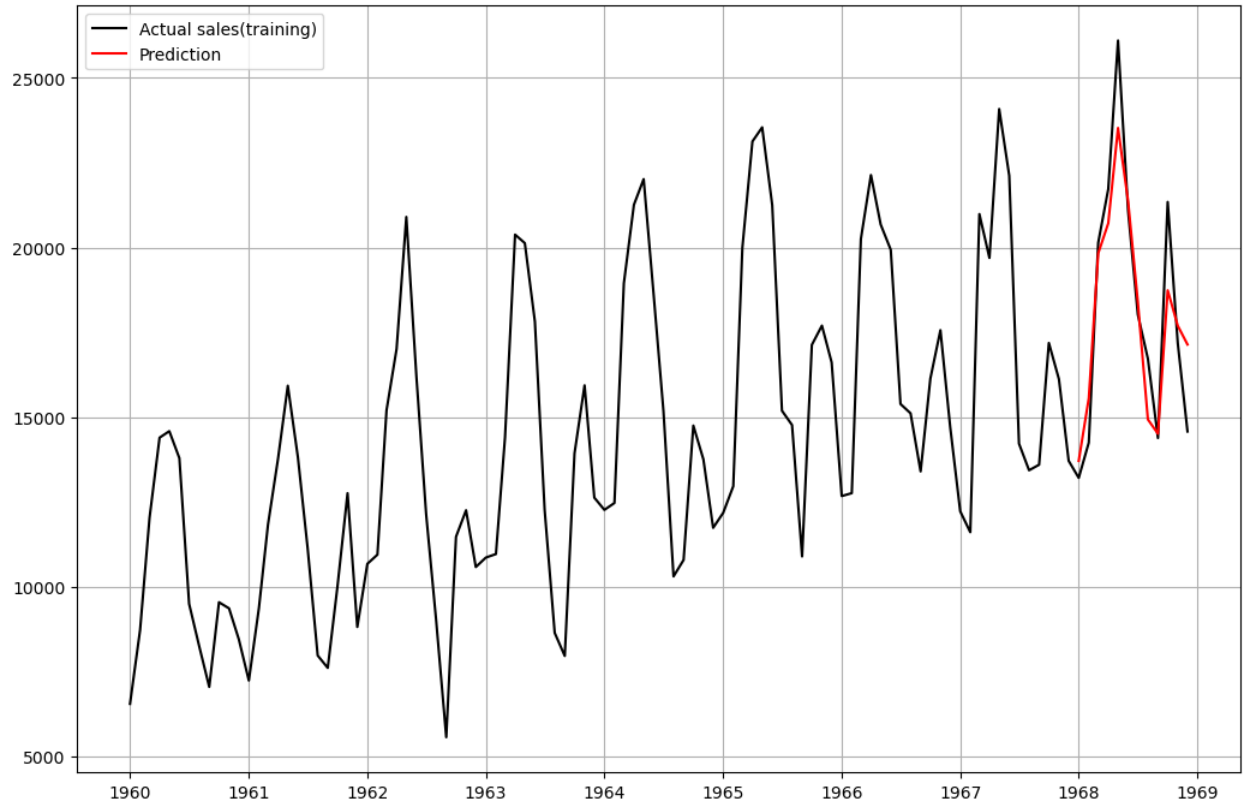

Customer class1
Customer class2
Customer class3
Customer class4
Customer class5
Customer class6
Age
20
30
40
50
60
70
Annual Income
20
40
60
80
100
120
140
Spending Score
0
20
40
60
80
100

Tehtävä 16.


1e14
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Inertia
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Number of clusters

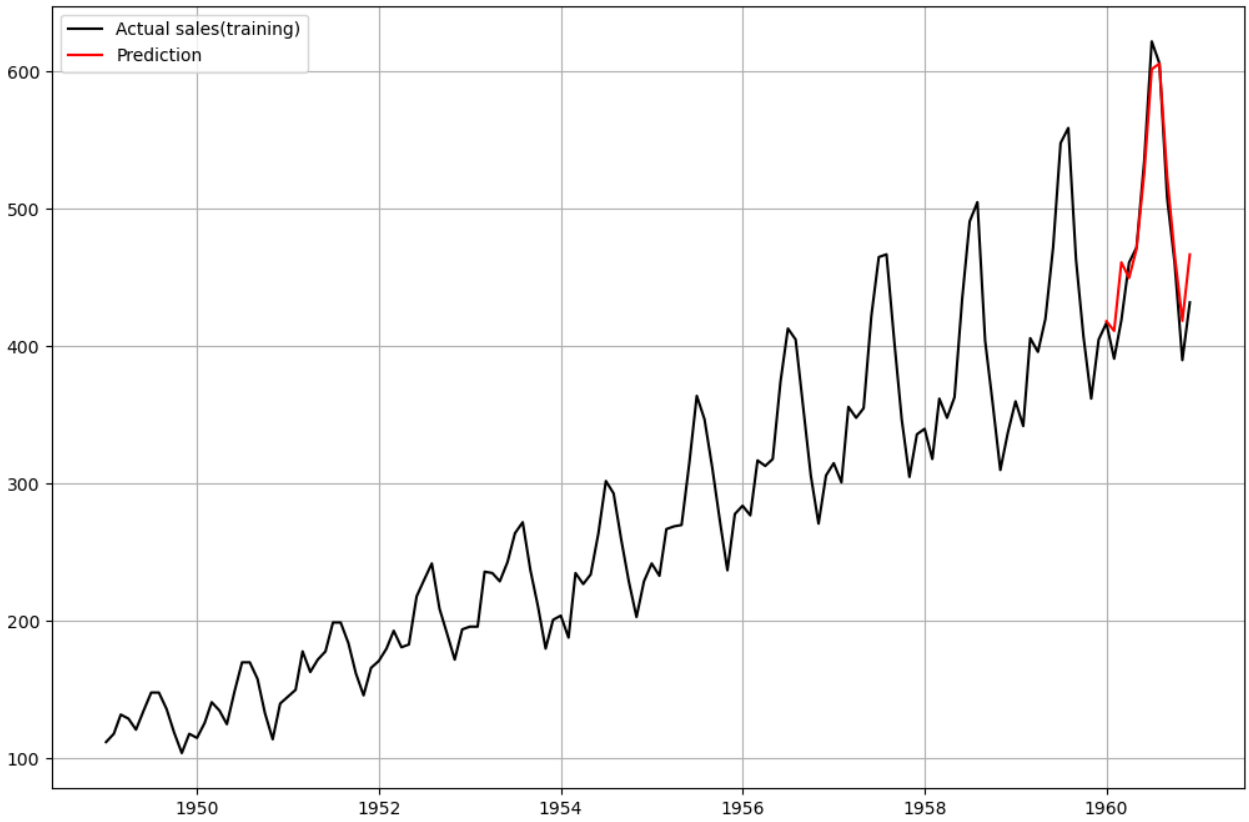

1e6
1.0
0.8
0.6
0.4
0.2
0.0
Y
200000
400000
600000
800000
X
Class1
Class2
Class3
Class4
Class5
Class6
Class7

Tehtävä 17.



Mean absolute error: 1163.2

Tehtävä 18.



Mean absolute error: 15.8

Tehtävä 19.

Käytettiin Kerasin luokittelevaa neuroverkkomallia (sequential). Kategorinen tarkkuus 0.975.

sample20 - DataFrame

Index	Machine ID	Team	Provider	Lifetime	PressureInd	MoistureIn	nperatureI	Broken	Ennuste
766	767	TeamB	Provider1	79	96.4868	96.4725	134.099	1	0.986
497	498	TeamB	Provider2	93	115.019	113.862	86.3152	1	0.999
358	359	TeamB	Provider1	80	104.703	101.73	68.59	1	0.986
569	570	TeamA	Provider2	17	94.6895	108.282	89.0581	0	0.0
585	586	TeamA	Provider3	65	112.894	104.743	107.361	1	0.994
66	67	TeamB	Provider4	39	87.3928	106.166	84.5217	0	0.0
938	939	TeamA	Provider3	65	119.001	108.567	90.2191	1	0.993
453	454	TeamC	Provider3	60	73.8268	82.7803	121.202	1	0.974
223	224	TeamA	Provider3	47	121.625	98.6943	74.7506	0	0.0
603	604	TeamA	Provider2	34	56.9187	74.7474	94.9234	0	0.0
221	222	TeamB	Provider3	65	104.641	111.419	96.5115	1	0.993
745	746	TeamB	Provider4	37	93.7078	114.397	115.083	0	0.0
94	95	TeamA	Provider3	52	86.1075	93.6841	83.157	0	0.003
360	361	TeamA	Provider4	63	90.9918	93.9535	81.0992	0	0.0
132	133	TeamB	Provider3	55	77.4666	89.2018	84.5391	0	0.109
365	366	TeamC	Provider2	29	80.7822	94.5538	87.4737	0	0.0
559	560	TeamA	Provider4	46	133.484	96.446	86.1177	0	0.0
491	492	TeamC	Provider1	59	98.0161	97.5377	99.204	0	0.0
417	418	TeamC	Provider4	41	118.514	96.8173	127.28	0	0.0
586	587	TeamA	Provider3	61	109.26	92.2502	70.1168	0	0.786

Index	Machine ID	Ennuste
919	920	0.983
562	563	0.971
595	596	0.925
192	193	0.916
901	902	0.915
372	373	0.909
350	351	0.906
639	640	0.887
25	26	0.872
163	164	0.863

Tehtävä 20.

Käytettiin Kerasin luokittelevaa neuroverkkomallia (sequential).

Test datan tarkkuus: 0.74

Training datan tarkkuus: 0.902

Index	churn	mated ch	churn risk
549	1	1	0.734
134	0	0	0.001
414	0	0	0.006
175	1	1	0.969
862	1	1	0.979
687	1	0	0.191
956	0	0	0.158
945	1	0	0.45
447	0	0	0.079
349	0	0	0.139
126	0	0	0.329
564	0	0	0.004
937	0	0	0.01
983	0	0	0.395
379	0	0	0.068
1	1	1	0.961
946	0	1	0.968
164	0	0	0.003
608	1	1	0.709
617	0	1	0.71

Lähdekoodit:

Tehtävä 1. a

```
"""
Created on Sun Nov  1 15:16:01 2020

@author: Sami
"""
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'])
X = X.reshape(-1,1) #tämä ohjeistettiin konsolissa
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'])
X_test = X_test.reshape(-1,1)
ennuste_test = model.predict(X_test)
df_test['Ennuste'] = ennuste_test

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

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

Tehtävä 1. b

```
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']])
# X = X.reshape(-1,1) #tämä ohjeistettiin konsolissa
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']])
# X_test = X_test.reshape(-1,1)
ennuste_test = model.predict(X_test)
df_test['Ennuste'] = ennuste_test

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

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

Tehtävä 2. a

```
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(7)
df_test = df[:185]
df_train = df[185:]

X = np.array(df_train[['Time', 'Close']])
# X = X.reshape(-1,1) #tämä ohjeistettiin konsolissa
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']])
# X_test = X_test.reshape(-1,1)
ennuste_test = model.predict(X_test)
df_test['Ennuste'] = ennuste_test

plt.scatter(df['Date'].values, df['Close'].values, color='black')
plt.plot((df_train['Date']+ pd.DateOffset(days=7)).values, df_train['Ennuste'].values, color='blue')
plt.plot((df_test['Date'] + pd.DateOffset(days=7)).values, df_test['Ennuste'].values, color='red')
plt.show()

df_validation = df_test.dropna()

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

error_train = mean_absolute_error(df_train['CloseFuture'], df_train['Ennuste'])
print("Ennusteen keskivirhe training datassa on %.f" % error_train)
```

Tehtävä 2. b

```python
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(60)
df_test = df[:185]
df_train = df[185:]

X = np.array(df_train[['Time', 'Close']])
# X = X.reshape(-1,1) #tämä ohjeistettiin konsolissa
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']])
# X_test = X_test.reshape(-1,1)
ennuste_test = model.predict(X_test)
df_test['Ennuste'] = ennuste_test

plt.scatter(df['Date'].values, df['Close'].values, color='black')
plt.plot((df_train['Date']+ pd.DateOffset(days=60)).values, df_train['Ennuste'].values, color='blue')
plt.plot((df_test['Date'] + pd.DateOffset(days=60)).values, df_test['Ennuste'].values, color='red')
plt.show()

df_validation = df_test.dropna()

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

error_train = mean_absolute_error(df_train['CloseFuture'], df_train['Ennuste'])
print("Ennusteen keskivirhe training datassa on %.f" % error_train)
```

Tehtävä 3.

```
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

a = {'x':[1,2,3,4,5], 'y':[1,2,1.3,3.75,2.25]}
df = pd.DataFrame(data = a)
print(df)
df['CloseFuture'] = df['y'].shift(1)
df_test = df[:5]
df_train = df[5:]


X = np.array(df['x'])
X = X.reshape(-1,1) #tämä ohjeistettiin konsolissa
y = np.array(df['y'])

model = linear_model.LinearRegression()

model.fit(X,y)
ennuste_train = model.predict(X)
df['Ennuste'] = ennuste_train
print(model.predict(X))

plt.scatter(df['x'].values, df['y'].values, color='black')
plt.plot((df['x']+1).values, df['Ennuste'].values, color='blue')

plt.show()

print("Ennuste muuttujan y arvoksi x:n arvolla 6 on", df['Ennuste'].values[4])
```

Tehtävä 4.

```
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 sklearn import preprocessing

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']])
scaler = preprocessing.MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y = np.array(df_train['CloseFuture'])

model = tf.keras.Sequential([
tf.keras.layers.Dense(10, activation="relu", input_shape=(2,)), #10
neuronia hidden layerillä, 2 input muuttujaa, bias termi rectified
linear funktio
tf.keras.layers.Dense(10, activation="relu"), #toinen piilotettu kerros
tf.keras.layers.Dense(1)])  # mallin output kerros

model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
```

```
                loss='mse', # numeerinen välimatka asteikollinen -> mse =
mean square error, jos luokitteluarvollinen output-muuttuja
categorical_crossentropy
                metrics=['mae'])  #mae, koska numeerinen mean acss error,
jos luokittelu -> accuracy
# kutsutaan myös back propagation algoritm

model.fit(X_scaled, y, epochs = 100, batch_size = 10) # epochs: montako
kertaa käydään läpi training data, batch_size: monenkodatarivin jälkeen
päivitetään painokertoimia
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')
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()

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

Tehtävä 5.

```
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 sklearn import preprocessing

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']])
X = X.reshape(-1,1)
scaler = preprocessing.MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y = np.array(df_train['CloseFuture'])

model = tf.keras.Sequential([
tf.keras.layers.Dense(20, activation="sigmoid", input_shape=(1,)), #10
neuronia hidden layerillä, 2 input muuttujaa, bias termi rectified
linear funktio
tf.keras.layers.Dense(20, activation="sigmoid"), #toinen piilotettu
kerros
tf.keras.layers.Dense(20, activation="relu"),
tf.keras.layers.Dense(1)])  # mallin output kerros

model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
```

```
                loss='mse', # numeerinen välimatka asteikollinen -> mse =
        mean square error, jos luokitteluarvollinen output-muuttuja
        categorical_crossentropy
                metrics=['mae'])  #mae, koska numeerinen mean absolute
        error, jos luokittelu -> accuracy
        # kutsutaan myös back propagation algoritm

        model.fit(X_scaled, y, epochs = 140, batch_size = 10) # epochs: montako
        kertaa käydään läpi training data, batch_size: monenkodatarivin jälkeen
        päivitetään painokertoimia
        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')
        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()

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

Tehtävä 6a.

```
        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/Kysynta.csv', sep=';', encoding='latin-1')
        df['Date'] = df['day']
        df['Time']= df.apply(lambda row: len(df) - row.name, axis=1)
        df['CloseFuture'] = df['demand'].shift(60)
        df_test = df[:240]
        df_train = df[240:]

        X = np.array(df_train[['Time']])
        X = X.reshape(-1,1)
        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']])
        # X_test = X_test.reshape(-1,1)
        ennuste_test = model.predict(X_test)
        df_test['Ennuste'] = ennuste_test

        plt.scatter(df['Date'].values, df['demand'].values, color='black')
        plt.plot(df_train['Date'].values+60, df_train['Ennuste'].values,
        color='blue')
```

```
plt.plot(df_test['Date'].values+60, df_test['Ennuste'].values,
color='red')
plt.show()

df_validation = df_test.dropna()

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

error_train = mean_absolute_error(df_train['CloseFuture'],
df_train['Ennuste'])
print("Ennusteen keskivirhe training datassa on %.f" % error_train)
```

Tehtävä 6b.

```
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 sklearn import preprocessing

df = pd.read_csv('data/Kysynta.csv', sep=';', encoding='latin-1')

df_test = df[300:]
df_train = 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'])

model = tf.keras.Sequential([
tf.keras.layers.Dense(10, activation="tanh", input_shape=(1,)),
tf.keras.layers.Dense(10, activation="tanh"),
tf.keras.layers.Dense(10, activation="relu"),
tf.keras.layers.Dense(1)])

model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss='mse',
              metrics=['mae'])

model.fit(X_scaled, y, epochs = 50, batch_size = 10, verbose = 2)

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=1)
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_test_validation = df_test.dropna()
df_train_validation = df_train.dropna()

error_train = mean_absolute_error(df_train_validation['Kysyntä'],
df_train_validation['Ennuste'])
print("Ennusteen keskivirhe training datassa on %.f" % error_train)

error = mean_absolute_error(df_test_validation['Kysyntä'],
df_test_validation['Ennuste'])
print("Ennusteen keskivirhe test datassa on %.f" % error)
```

Tehtävä 7.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.metrics import accuracy_score
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv('data/fruit_data.csv', sep=',', encoding='latin-1')

X = np.array(df[['mass', 'width', 'height', 'color_score']])

koodit = {'apple':0, 'lemon':1,'mandarin':2,'orange':3 }
df['fruit_name_koodi'] = df['fruit_name'].map(koodit)
y = np.array(df['fruit_name_koodi'])

scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X)

# Luodaan logistinen regressiomalli
model = linear_model.LogisticRegression(multi_class='multinomial',
solver='newton-cg') #multi_class ja solver lisänä koska luokiteltavia
muuttujia on neljä.
model.fit(X_scaled,y)
ennuste = model.predict(X_scaled)
print("Logistisen regressiomallin tarkkuus: " ,accuracy_score(y,
ennuste))
df['LRennuste'] = ennuste

# Luodaan SVM malli
model_SVC = SVC()
model_SVC.fit(X_scaled,y)
ennuste_SVC = model_SVC.predict(X_scaled)
print("SVM mallin tarkkuus: ",accuracy_score(y, ennuste_SVC))
df['SVCennuste'] = ennuste_SVC

# Luodaan KNN malli
model_KNN = KNeighborsClassifier()
model_KNN.fit(X_scaled,y)
ennuste_KNN = model_KNN.predict(X_scaled)
print("KNN mallin tarkkuus: ",accuracy_score(y, ennuste_KNN))
df['KNNennuste'] = ennuste_KNN
```

Tehtävä 8.

```
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/fruit_data.csv', sep=',', encoding='latin-1')

X = np.array(df[['mass', 'width', 'height', 'color_score']])

y = np.array(pd.get_dummies(df['fruit_name']))

scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X)

model = tf.keras.Sequential([
tf.keras.layers.Dense(30, activation="relu",
input_shape=(X_scaled.shape[1],)),  #X_scaled.shape[1] arvo on 4
tf.keras.layers.Dense(30, activation="relu"),
tf.keras.layers.Dense(4, activation=tf.nn.softmax) # luokittelevassa
neuroverkossa täytyy olla yhtä monta output muuttujaa kuin on output
luokkia eli tässä 4
]) # luokittelevassa neuroverkossa outputkerroksen activointifunktioksi
määritellään softmax

model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])

model.fit(X_scaled, y, epochs=50, batch_size=1)

ennuste = np.argmax(model.predict(X_scaled), axis=1)
df['Ennuste'] = ennuste
```

Tehtävä 9.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import linear_model
from sklearn.metrics import accuracy_score

df = pd.read_csv('data/Titanic.csv', sep=',', encoding='latin-1')

df_test = df.sample(n = 200)
df_train = df.drop(df_test.index)

X = (pd.get_dummies(df_train[['Pclass', 'Sex', 'SibSp','Embarked']]))
X['Age'] = df_train['Age']
X['Age'] = X['Age'].fillna(0)
X['Parch'] = df_train['Parch']
X = np.array(X)

y = np.array(df_train['Survived'])

scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X)

# =======================================================
# ------ Luodaan logistinen regressiomalli ---------------
```

```
model = linear_model.LogisticRegression(multi_class='multinomial',
solver='newton-cg') #multi_class ja solver lisänä koska luokiteltavia
muuttujia on neljä.
model.fit(X_scaled,y)
ennuste = model.predict(X_scaled)
print("Logistisen regressiomallin tarkkuus (training data): " ,
round(accuracy_score(y, ennuste)*100,2),"%")
df_train['LRennuste'] = ennuste
# -------------------------------------------------------

# --------- LR TEST -------------------
X_test = (pd.get_dummies(df_test[['Pclass', 'Sex',
'SibSp','Embarked']]))
X_test['Age'] = df_test['Age']
X_test['Age'] = X_test['Age'].fillna(0)
X_test['Parch'] = df_test['Parch']

X_test = np.array(X_test)
y_test = np.array(df_test['Survived'])

X_testscaled = scaler.transform(X_test)
ennuste_test = model.predict(X_testscaled)
df_test['LREnnuste'] = ennuste_test
print("Logistisen regressiomallin tarkkuus test datassa: "
,round(accuracy_score(y_test, ennuste_test)*100,2),"%")
# ---------------------------------

# =======================================================
# ----------- Luodaan SVM malli --------------------------
model_SVC = SVC()
model_SVC.fit(X_scaled,y)
ennuste_SVC = model_SVC.predict(X_scaled)
print("SVM mallin tarkkuus (training data): ", round(accuracy_score(y,
ennuste_SVC)*100,2),"%")
df_train['SVMennuste'] = ennuste_SVC
# -------------------------------------------------------

# --------- SVM TEST -------------------
X_test = (pd.get_dummies(df_test[['Pclass', 'Sex',
'SibSp','Embarked']]))
X_test['Age'] = df_test['Age']
X_test['Age'] = X_test['Age'].fillna(0)
X_test['Parch'] = df_test['Parch']

X_test = np.array(X_test)
y_test = np.array(df_test['Survived'])

X_testscaled = scaler.transform(X_test)
ennuste_test = model_SVC.predict(X_testscaled)
df_test['SVMEnnuste'] = ennuste_test
print("SVM tarkkuus test datassa: " ,round(accuracy_score(y_test,
ennuste_test)*100,2),"%")
# ---------------------------------

# =======================================================
# ------------------- Luodaan KNN malli ------------------
model_KNN = KNeighborsClassifier()
model_KNN.fit(X_scaled,y)
ennuste_KNN = model_KNN.predict(X_scaled)
print("KNN mallin tarkkuus (training data): ", round(accuracy_score(y,
ennuste_KNN)*100,2),"%")
df_train['KNNennuste'] = ennuste_KNN
# -------------------------------------------------------
```

```
# --------- KNN TEST -------------------
X_test = (pd.get_dummies(df_test[['Pclass', 'Sex',
'SibSp','Embarked']]))
X_test['Age'] = df_test['Age']
X_test['Age'] = X_test['Age'].fillna(0)
X_test['Parch'] = df_test['Parch']

X_test = np.array(X_test)
y_test = np.array(df_test['Survived'])

X_testscaled = scaler.transform(X_test)
ennuste_test = model_KNN.predict(X_testscaled)
df_test['KNNEnnuste'] = ennuste_test
print("KNN tarkkuus test datassa: " ,round(accuracy_score(y_test,
ennuste_test)*100,2),"%")
# ---------------------------------

df_otos = df_test[['PassengerId','Survived','SVMEnnuste']].sample(n=20)
```

Tehtävä 10.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import linear_model
from sklearn.metrics import accuracy_score

df = pd.read_csv('data/Titanic.csv', sep=',', encoding='latin-1')

df_test = df.sample(n = 200)
df_train = df.drop(df_test.index)

X = (pd.get_dummies(df_train[['Pclass', 'Sex', 'SibSp','Embarked']]))
X['Age'] = df_train['Age']
X['Age'] = X['Age'].fillna(0)
X['Parch'] = df_train['Parch']
X = np.array(X)

y = np.array(pd.get_dummies(df_train['Survived']))

scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X)

# ========================================================
model = tf.keras.Sequential([
tf.keras.layers.Dense(30, activation= tf.nn.relu ,
input_shape=(X_scaled.shape[1],)),  # input muuttujien lkm
tf.keras.layers.Dense(30, activation= tf.nn.relu ),
tf.keras.layers.Dense(2, activation=tf.nn.softmax) # luokittelevassa
neuroverkossa täytyy olla yhtä monta output muuttujaa kuin on output
luokkia eli tässä 4
]) # luokittelevassa neuroverkossa outputkerroksen activointifunktioksi
määritellään softmax

model.compile(optimizer=tf.keras.optimizers.Adam(0.005),
              loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])
```

```
model.fit(X_scaled, y, epochs=20, batch_size=1)

ennuste = np.argmax(model.predict(X_scaled), axis=1)
df_train['Ennuste'] = ennuste

# --------- LR TEST -------------------
X_test = (pd.get_dummies(df_test[['Pclass', 'Sex',
'SibSp','Embarked']]))
X_test['Age'] = df_test['Age']
X_test['Age'] = X_test['Age'].fillna(0)
X_test['Parch'] = df_test['Parch']

X_test = np.array(X_test)
y_test2 = np.array(df_test['Survived'])
y_test = np.array(pd.get_dummies(df_test['Survived']))

X_testscaled = scaler.transform(X_test)
ennuste_test = np.argmax(model.predict(X_testscaled),axis=1)
df_test['Ennuste'] = ennuste_test
print("tarkkuus test datassa: ",round(accuracy_score(y_test2,
ennuste_test)*100,2),"%")
# ---------------------------------

df_otos = df_test[['PassengerId','Survived','Ennuste']].sample(n=20)
```

Tehtävä 11.

```
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() # haetaan kerasin valmis datasetti
mnist

#%%
# tehdään x_train ja x_test muuttujien koosta yksiulotteinen taulukko
kertomalla 28x28=784
x_train_flat=x_train.reshape(60000, 784)
x_test_flat = x_test.reshape(10000, 784)

x_train_flat = x_train_flat/255 # jokainen pikseli on välillä 0-255,
joten jaetaan 255 ja saadaan luku välillä 0-1
x_test_flat = x_test_flat/255

y_train = pd.get_dummies(y_train)
y_test_orig = y_test
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'),
    tf.keras.layers.Dense(10, activation='softmax') #outputissa 10
numeroa ja luokittelevan neuroverkon output activation on softmax
    ])

model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='categorical_crossentropy',
              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)

enn = pd.DataFrame(ennuste_test)
enn['max'] = enn.max(axis=1)
enn['ennuste'] = enn.idxmax(axis=1)
enn['oikea'] = y_test_orig
enn['tulos'] = enn['ennuste'] == enn['oikea']
#%%
import random

fig, axs = plt.subplots(2, 3)
axs = axs.ravel()
x=0
lista = [115,124,149,151,247,3893]
for i in lista:
    # i = random.randint(0,len(x_test))
    # number = y_test.loc[i][lambda x: x == 1].index[0]
    number = y_test_orig[i]
    #print(number)
    # pred_number = ennuste_test[i].max()
    pred_number = enn['ennuste'][i]

    print(pred_number)
    axs[x].imshow(x_test[i])
    axs[x].text(27,-1, i, size=9, ha="right")
    axs[x].text(0,-1, ('pr',pred_number), size=9 )
    axs[x].text(12,-1, number, size=9 )
    axs[x].set_xticks([])
    axs[x].set_yticks([])
    x+=1
```

Tehtävä 12.

```
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() # haetaan kerasin valmis datasetti
mnist


#%%
# tehdään x_train ja x_test muuttujien koosta yksiulotteinen taulukko
kertomalla 28x28=784
x_train_flat=x_train.reshape(60000, 28, 28, 1) #nro 1 merkitsee kuvan
mustavalkoiseksi. RGB kuvassa olisi kolme 28px taulukkoa
x_test_flat = x_test.reshape(10000, 28, 28 ,1)

x_train_flat = x_train_flat/255 # jokainen pikseli on välillä 0-255,
joten jaetaan 255 ja saadaan luku välillä 0-1
x_test_flat = x_test_flat/255
y_test_orig = y_test

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)), #2-ulotteinen konvoluutiokerros, kunkin ikkunan
koko 5x5px. Ei laitettu strides, joten ikkunat menevät tässä limittäin.
    tf.keras.layers.MaxPooling2D(pool_size=2, strides=2), #ikkunan koko
2x2px ja siirretään eteenpäin 2px. Ikkunat eivät siis mene limittäin.
    tf.keras.layers.Conv2D(15, kernel_size=5, activation='relu'), #15
filtteriä
    tf.keras.layers.MaxPooling2D(pool_size=2, strides=2),
    tf.keras.layers.Flatten(),  #konvoluutio osuus loppuu
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax') #outputissa 10
numeroa ja luokittelevan neuroverkon output activation on softmax
    ])

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

model.fit(x_train_flat, y_train, validation_data=(x_test_flat, y_test),
epochs=11, batch_size=100)

#%%
ennuste_test = model.predict(x_test_flat)
enn = pd.DataFrame(ennuste_test)
enn['max'] = enn.max(axis=1)
enn['ennuste'] = enn.idxmax(axis=1)
enn['oikea'] = y_test_orig
enn['tulos'] = enn['ennuste'] == enn['oikea']

#%%
fig, axs = plt.subplots(2, 3)
axs = axs.ravel()
x=0
lista = [115,124,149,151,247,3893]
for i in lista:
    # i = random.randint(0,len(x_test))
    # number = y_test.loc[i][lambda x: x == 1].index[0]
    number = y_test_orig[i]
    #print(number)
    # pred_number = ennuste_test[i].max()
    pred_number = enn['ennuste'][i]

    print(pred_number)
    axs[x].imshow(x_test[i])
    axs[x].text(27,-1, i, size=9, ha="right")
    axs[x].text(0,-1, ('pr',pred_number), size=9 )
    axs[x].text(12,-1, number, size=9 )
    axs[x].set_xticks([])
    axs[x].set_yticks([])
    x+=1

#%%
model.fit(x_train_flat, y_train, validation_data=(x_test_flat, y_test),
epochs=1, batch_size=100)
```

Tehtävä 13.

```
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import cv2
from pathlib import Path
import random
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator

folder = Path("/Users/Sami/Downloads/dogs-vs-cats/")
path_train = Path("/Users/Sami/Downloads/dogs-vs-cats/train/")
path_test = Path("/Users/Sami/Downloads/dogs-vs-cats/test1/")
path_array = []
label=[]

test_array = []
test_label = []

for file in os.listdir(path_train):
    path_array.append(os.path.join(path_train,file))
    if file.startswith("cat"):
        label.append('cat')
    elif file.startswith("dog"):
        label.append('dog')

print(path_array[:5])
print(label[:5])

d = {'path': path_array, 'label': label}
X_train = pd.DataFrame(data=d)
# X_train.head()

for file in os.listdir(path_test):
    test_array.append(os.path.join(path_test,file))


test_d = {'path': test_array}
X_test = pd.DataFrame(data=test_d)


#%%
model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16, (3,3),
activation='relu', input_shape=(96,96,3)),
                                    tf.keras.layers.MaxPooling2D(2,2),
                                    tf.keras.layers.Conv2D(32, (3,3),
activation='relu'),
                                    tf.keras.layers.MaxPooling2D(2,2),
                                    tf.keras.layers.Conv2D(64, (3,3),
activation='relu'),
                                    tf.keras.layers.MaxPooling2D(2,2),
                                    tf.keras.layers.Flatten(),
                                    tf.keras.layers.Dense(64,
activation='relu'),
                                    tf.keras.layers.Dense(64,
activation='relu'),
                                    tf.keras.layers.Dense(2,
activation='softmax')
])
model.compile(optimizer=tf.keras.optimizers.RMSprop(0.001),
loss='categorical_crossentropy', metrics=['acc'])
model.summary()
```

```
#%%
dim = (96,96)
train_datagen = ImageDataGenerator(validation_split=0.08 ,rescale =
1.0/255.)
train_generator = train_datagen.flow_from_dataframe(dataframe=
X_train,x_col='path',y_col='label',subset="training",batch_size=50,seed
=42,shuffle=True, class_mode= 'categorical', target_size = dim)
valid_generator = train_datagen.flow_from_dataframe(dataframe=
X_train,x_col='path',y_col='label',subset="validation",batch_size=50,se
ed=42,shuffle=True, class_mode= 'categorical', target_size = dim)


#%%

STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size

fitted_model = model.fit_generator(generator=train_generator,
                    steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,
                    validation_steps=STEP_SIZE_VALID,
                    epochs=9,
                    verbose=1
)

#%%
accuracy = fitted_model.history['acc']
#plt.imshow(X_train[1])
print(fitted_model.history)
plt.plot(range(len(accuracy)), accuracy, 'bo', label = 'accuracy')
plt.legend()


#%%
from tensorflow.keras.preprocessing import image

def catOrDog(img_number):
    test_image = image.load_img(X_train['path'][img_number],
target_size=(96,96))
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis = 0)
    result = model.predict_classes(x=test_image)[0]
    prob = model.predict_proba(x=test_image)
    fitted_model.history
    print(prob)

    if result == 1:
        prediction = ('Dog (Pr: ' + str( np.round( (prob[0][1]),3) )
+')')

    else:
        prediction = ('Cat (Pr: ' + str( np.round( (prob[0][0]),3) )
+')')
    return prediction

def img_load(img_number):
    test_image = image.load_img(X_test['path'][img_number],
target_size=(150,150))
    return test_image


fig, axs = plt.subplots(3, 3)
axs = axs.ravel()
```

```
for x in range(9):
    i = random.randint(0,len(X_test))
    print(i)
    axs[x].imshow(img_load(i))
    axs[x].text(150,-5, i, size=9, ha="right")
    axs[x].text(0,-5, catOrDog(i), size=9 )


img_list = {50,4163,1300,508,11490,5375}
fig2, axs = plt.subplots(2, 3)
axs = axs.ravel()
i=0
for img in img_list:
    print(i)
    axs[i].imshow(img_load(img))
    axs[i].text(150,-5, img , size=9, ha="right")
    axs[i].text(0,-5, catOrDog(img), size=9)
    i+=1
```

Tehtävä 14.

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

df = pd.read_csv('data/Mall_Customers.csv')

X = np.array(df[['Age', 'Annual Income (k$)', 'Spending Score (1-
100)']])

model = KMeans(n_clusters=4)
model.fit(X)
labels = model.labels_
df['Label'] = labels

#%%
from mpl_toolkits.mplot3d import Axes3D

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]['Age'].values
    y=df.loc[df['Label'] == i]['Annual Income (k$)'].values
    z=df.loc[df['Label'] == i]['Spending Score (1-100)'].values
    ax.scatter(x,y,z,marker='o',s=40, color=colors[i],label='Customer
class'+str(i+1))
ax.set_xlabel('Age')
ax.set_ylabel('Annual Income')
ax.set_zlabel('Spending Score')
ax.legend()
plt.show()
```

Tehtävä 15.

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

df = pd.read_csv('data/Mall_Customers.csv')

X = np.array(df[['Age', 'Annual Income (k$)', 'Spending Score (1-
100)']])

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


#%%
from mpl_toolkits.mplot3d import Axes3D

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]['Age'].values
    y=df.loc[df['Label'] == i]['Annual Income (k$)'].values
    z=df.loc[df['Label'] == i]['Spending Score (1-100)'].values
    ax.scatter(x,y,z,marker='o',s=40, color=colors[i],label='Customer
class'+str(i+1))
ax.set_xlabel('Age')
ax.set_ylabel('Annual Income')
ax.set_zlabel('Spending Score')
ax.legend()
plt.show()
```

Tehtävä 16.

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

df = pd.read_csv('data/2dclusters.csv', sep=';', header=None,
names=['A', 'B'])

#%%
X = np.array(df[['A','B']])

inertia = []
for i in range(1,19):
    model = KMeans(n_clusters=i)
    model.fit(X)
    inertia.append(model.inertia_)

plt.scatter(np.arange(1,19),inertia)
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

#%%
model = KMeans(n_clusters=7)
model.fit(X)
labels = model.labels_
df['Label'] = labels

#%%
from mpl_toolkits.mplot3d import Axes3D

colors = {0:'red', 1:'blue', 2:'green', 3:'magenta', 4:'black',
5:'orange', 6:'yellow'}

fig = plt.figure()
ax = fig.add_subplot(111)

for i in range(0,7):
    x=df.loc[df['Label'] == i]['A'].values
    y=df.loc[df['Label'] == i]['B'].values

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

ax.legend()
plt.show()
```

Tehtävä 17.

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

ennusteaika = 12 #kuukautta ennustetaan myyntiä 12kk eteenpäin
seqlength = 12 # syöteverktorin (historia) pituus kuukausina. Eli 12kk
myynnit annetaan syötteenä, vapaaval

df = pd.read_csv('data/monthly-car-sales.csv')
df['Month'] = pd.to_datetime(df['Month'])
```

```
df['Time'] = df.index

#%%
# stationaarinen "muutos"-aikasarja. HUOM trendi katoaa
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,ennusteaika+1):
    df['SalesDiffFut'+str(i)] = df['SalesDiff'].shift(-i)

df_train = df.iloc[:-2*ennusteaika]
df_train.dropna(inplace=True)
df_test = df.iloc[-2*ennusteaika:]

#%% Muuttujien valinta ja skaalaus
input_vars = ['SalesDiff']
for i in range(1,seqlength):
    input_vars.append('SalesDiffFut'+str(i))

output_vars = []
for i in range(1, ennusteaika+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)  # viimeinen
ykkönen kertoo, että on yksi aikasarja josta ennustetaan
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)
#LSTM vaatima muoto

#%%
# trendin mallinnus lineaarisella regressiolla
from sklearn import linear_model
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)

#%%

slope = modelLR.coef_

#%% LSTM-verkon muodostus ja opetus
modelLSTM = tf.keras.Sequential([
    tf.keras.layers.LSTM(24, input_shape=(seqlength,1), # input vain
yksi aikasarja, mutta voisi olla myös useampia
```

```
                            return_sequences=False), #palautetaanko
kerroksen outputtiin kunkin LSTM solun piilotettu tila vai ei
    # tf.keras.layers.LSTM(24, return_sequences=False), voisi olla
toinenkin kerros, jolloin return_seq edellisessä olisi true
    tf.keras.layers.Dense(ennusteaika)
    ])
modelLSTM.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='mse',
              metrics=['mae'])

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

#%% Ennusteen määritys. huom ennuste = ennusteDiff+ trendi
ennusteDiff = scalero.inverse_transform(
    modelLSTM.predict(X_testscaledLSTM[ennusteaika-1].reshape(1,12,1)))
ennuste = np.zeros(13)
ennuste[0] = df_test['Sales'][df_test.index[ennusteaika-1]]
for i in range(1,13):
    for j in range(1,13):
        ennuste[j] = ennuste[j-1]+ennusteDiff[0][j-1]+slope
#ennuste=ed.kk myynti+muutos+vuositrendin ka muutos
ennuste = np.array(ennuste[1:])

#%% Luodaan ennusteen piirtämistä varten oma dataframe
df_pred = df_test[-12:]
df_pred['SalesPred'] = ennuste
#%% Piirretään pyydetty kuvaaja
plt.plot(df['Month'].values, df['Sales'].values, color='black',
label='Actual sales(training)')
plt.plot(df_pred['Month'].values, df_pred['SalesPred'], color='red',
label='Prediction')

plt.grid()
plt.legend()
plt.show()
#%%
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(df_pred['Sales'].values,
                          df_pred['SalesPred'].values))
```

Tehtävä 18.

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

ennusteaika = 12 #kuukautta ennustetaan myyntiä 12kk eteenpäin
seqlength = 12 # syöteverktorin (historia) pituus kuukausina. Eli 12kk
myynnit annetaan syötteenä, vapaaval

df = pd.read_csv('data/AirPassengers.csv')
df['Month'] = pd.to_datetime(df['Month'])
df['Time'] = df.index

#%%
# stationaarinen "muutos"-aikasarja. HUOM trendi katoaa
df['PssLag'] = df['Passengers'].shift(1)
df['PssDiff']= df.apply(lambda row:
                          row['Passengers']-row['PssLag'], axis=1)

for i in range(1, seqlength):
    df['PssDiffLag'+str(i)] = df['PssDiff'].shift(i)
```

```
for i in range(1,ennusteaika+1):
    df['PssDiffFut'+str(i)] = df['PssDiff'].shift(-i)

df_train = df.iloc[:-2*ennusteaika]
df_train.dropna(inplace=True)
df_test = df.iloc[-2*ennusteaika:]

#%% Muuttujien valinta ja skaalaus
input_vars = ['PssDiff']
for i in range(1,seqlength):
    input_vars.append('PssDiffFut'+str(i))

output_vars = []
for i in range(1, ennusteaika+1):
    output_vars.append('PssDiffFut'+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)  # viimeinen
ykkönen kertoo, että on yksi aikasarja josta ennustetaan
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)
#LSTM vaatima muoto

#%% trendin mallinnus lineaarisella regressiolla
from sklearn import linear_model
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['PassengerAvgPred'] = modelLR.predict(XLR_test)

#%%
slope = modelLR.coef_

#%% LSTM-verkon muodostus ja opetus
modelLSTM = tf.keras.Sequential([
    tf.keras.layers.LSTM(24, input_shape=(seqlength,1), # input vain
yksi aikasarja, mutta voisi olla myös useampia
                         return_sequences=False), #palautetaanko
kerroksen outputtiin kunkin LSTM solun piilotettu tila vai ei
    # tf.keras.layers.LSTM(24, return_sequences=False), voisi olla
toinenkin kerros, jolloin return_seq edellisessä olisi true
    tf.keras.layers.Dense(ennusteaika)
    ])
modelLSTM.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='mse',
              metrics=['mae'])

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

#%% Ennusteen määritys. huom ennuste = ennusteDiff+ trendi
ennusteDiff = scalero.inverse_transform(
```

```
    modelLSTM.predict(X_testscaledLSTM[ennusteaika-1].reshape(1,12,1)))
ennuste = np.zeros(13)
ennuste[0] = df_test['Passengers'][df_test.index[ennusteaika-1]]
for i in range(1,13):
    for j in range(1,13):
        ennuste[j] = ennuste[j-1]+ennusteDiff[0][j-1]+slope
#ennuste=ed.kk myynti+muutos+vuositrendin ka muutos
ennuste = np.array(ennuste[1:])

#%% Luodaan ennusteen piirtämistä varten oma dataframe
df_pred = df_test[-12:]
df_pred['PssPred'] = ennuste
#%% Piirretään pyydetty kuvaaja
plt.plot(df['Month'].values, df['Passengers'].values, color='black',
label='Actual sales(training)')
plt.plot(df_pred['Month'].values, df_pred['PssPred'], color='red',
label='Prediction')

plt.grid()
plt.legend()
plt.show()
#%%
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(df_pred['Passengers'].values,
                          df_pred['PssPred'].values))
```

Tehtävä 19.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import mean_squared_error
#%% Luodaan ennustemalli
df = pd.read_csv('data/MachineData.csv', sep=';', decimal='.')

X = np.array(pd.get_dummies(df[['Machine
ID','Team','Provider','Lifetime','PressureInd','MoistureInd','Temperatu
reInd']]))
y = np.array(pd.get_dummies(df['Broken']))

scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X)

model = tf.keras.Sequential([
tf.keras.layers.Dense(20, activation="relu",
input_shape=(X_scaled.shape[1],)),  #X_scaled.shape[1] arvo on 4
tf.keras.layers.Dense(20, activation="relu"),
tf.keras.layers.Dense(2, activation=tf.nn.softmax) # luokittelevassa
neuroverkossa täytyy olla yhtä monta output muuttujaa kuin on output
luokkia eli tässä 4
]) # luokittelevassa neuroverkossa outputkerroksen activointifunktioksi
määritellään softmax

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

model.fit(X_scaled, y, epochs=10, batch_size=1)
#%%
ennuste = model.predict(X_scaled)
df['Ennuste'] = np.round(ennuste[:,1],3)
```

```
df.sort_values(by=['Ennuste'], ascending=False, inplace=True)

#%%
rikkoutuvat = df[df['Broken']==0]
jees = rikkoutuvat.nlargest(10, 'Ennuste', keep='first')

top10 = jees[['Machine ID', 'Ennuste']]

#%%
sample20 = df.sample(20)
```

Tehtävä 20.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import mean_squared_error

#%% Luodaan ennustemalli
df = pd.read_csv('data/Telco.csv', sep=';')
df.fillna(0, inplace=True)
df_test = df.sample(n = 100)
df_train = df.drop(df_test.index)

cols =
['tenure','age','address','ed','equip','ebill','internet','employ','log
card','callcard']

X_train = np.array(df_train[cols]).astype('int')
y_train = np.array(pd.get_dummies(df_train['churn']))

#%% Model
scaler = preprocessing.StandardScaler() # skaalaa
normaalijakautuneeksi. Vaatimus joissakin malleissa
X_scaled = scaler.fit_transform(X_train)

model = tf.keras.Sequential([
tf.keras.layers.Dense(24, activation="relu",
input_shape=(X_scaled.shape[1],)),  #X_scaled.shape[1] arvo on 4
tf.keras.layers.Dense(24, activation="relu"),
tf.keras.layers.Dense(16, activation="sigmoid"),
tf.keras.layers.Dense(2, activation="softmax") # luokittelevassa
neuroverkossa täytyy olla yhtä monta output muuttujaa kuin on output
luokkia eli tässä 4
]) # luokittelevassa neuroverkossa outputkerroksen activointifunktioksi
määritellään softmax

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

model.fit(X_scaled, y_train, epochs=50, batch_size=1)

ennuste = model.predict(X_scaled)
df_train['Ennuste'] = np.round(ennuste[:,1],0).astype(int)

#%% TEST

X_test = np.array(df_test[cols])
X_test_scaled = scaler.fit_transform(X_test)
ennuste_test = model.predict(X_test_scaled)
df_test['Estimated churn'] = np.round(ennuste_test[:,1],0).astype(int)
```

```
df_test['churn risk'] = np.round(ennuste_test[:,1],3)
y_test = np.array(df_test['Estimated churn'])

from sklearn.metrics import accuracy_score
print("Test datan tarkkuus: ", accuracy_score(y_test,
df_test['churn']))
print("Training datan tarkkuus: ",
np.round(accuracy_score(df_train['churn'], df_train['Ennuste']),3))
sample = df_test[['churn','Estimated churn','churn risk']].sample(20)
```