**TEHTÄVÄ1**

**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("/Users/kayttaja/.spyder-py3/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)

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"], 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()

print("Ennusteen keskivirhe test datassa on %.f"%

mean\_absolute\_error(df\_validation["CloseFuture"],

df\_validation["Ennuste"]))

****

**﻿**

**Ennusteen keskivirhe test datassa on 70**

**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("/Users/kayttaja/.spyder-py3/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"], 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()

print("Ennusteen keskivirhe test datassa on %.f"%

mean\_absolute\_error(df\_validation["CloseFuture"],

df\_validation["Ennuste"]))

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

****

**﻿Ennusteen keskivirhe test datassa on 35**

**Mallin kertoimet ovat**

**[0.1732308 0.58764722]**

**160.71598367544226**

**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("/Users/kayttaja/.spyder-py3/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,)),

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

tf.keras.layers.Dense(1)

])

model.compile(optimizer=tf.compat.v1.train.AdamOptimizer(0.001),

loss="mse",

metrics=["mae"])

model.fit(X\_scaled,y, epochs=10 , 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"], 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()

print("Ennusteen keskivirhe test datassa on %.f"%

mean\_absolute\_error(df\_validation["CloseFuture"],

df\_validation["Ennuste"]))

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



**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 tensorflow import keras

#from tensorflow.keras import layers

from sklearn import preprocessing

df=pd.read\_csv("/Users/kayttaja/.spyder-py3/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(10, activation='relu', input\_shape=(1,)),

tf.keras.layers.Dense(10, activation='sigmoid', input\_shape=(1,)),

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

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

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

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

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

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

tf.keras.layers.Dense(1)

])

model.compile(#optimizer=tf.compat.v1.train.AdamOptimizer(0.001),

optimizer=tf.compat.v1.train.AdamOptimizer(0.01),

loss="mse",

metrics=["mae"])

model.fit(X\_scaled,y, epochs=30 , batch\_size=10 )

#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"], 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()

print("Ennusteen keskivirhe test datassa on %.f"%

mean\_absolute\_error(df\_validation["CloseFuture"],

df\_validation["Ennuste"]))



﻿Ennusteen keskivirhe test datassa on 100



﻿Ennusteen keskivirhe test datassa on 104

**TEHTÄVÄ 7**

﻿import pandas as pd

import numpy as np

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("/Users/kayttaja/.spyder-py3/fruit\_data.csv")

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

koodit={'apple':1,'lemon':0,'mandarin':2,'orange':3}

df['fruit\_label']=df['fruit\_name'].map(koodit)

y=df['fruit\_label']

scaler=preprocessing.StandardScaler()

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

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

model=SVC()

model.fit(X\_scaled,y)

ennuste=model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df['SVMennuste']=ennuste

model=KNeighborsClassifier()

model.fit(X\_scaled,y)

ennuste=model.predict(X\_scaled)

print(accuracy\_score(y, ennuste))

df['KNNennuste']=ennuste

df1=df[['fruit\_label','fruit\_name','fruit\_subtype',"mass","width","height","color\_score"]].head()

df1

**metin, skorbord içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**﻿0.8813559322033898**

**0.9661016949152542**

**0.9830508474576272**

**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("/Users/kayttaja/.spyder-py3/fruit\_data.csv")

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

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

scaler=preprocessing.StandardScaler()

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

model = keras.Sequential([

#tf.keras.Input(shape=(2,)

keras.layers.Dense(10, activation=tf.nn.relu,

input\_shape=(X\_scaled.shape[1],)),

#tf.keras.layers.Dense(10, activation='sigmoid', input\_shape=(1,)),

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

keras.layers.Dense(10, activation=tf.nn.relu),

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

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

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

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

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

])

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

optimizer=tf.compat.v1.train.AdamOptimizer(learning\_rate=0.001),

metrics =['categorical\_accuracy'])

model.fit(X\_scaled,y, epochs=10 , batch\_size=1)

#model.fit(X\_scaled,y, epochs=50 , batch\_size=1)

#ennuste=model.predict(X\_scaled)

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

df["Ennuste"]= ennuste

**metin, skorbord içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**﻿** **﻿categorical\_accuracy: 0.1017**

**TEHTÄVÄ 11**

﻿import tensorflow as tf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

#%%

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

#%%

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

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

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

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

y\_train=np.array(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'),

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

])

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

optimizer=tf.compat.v1.train.AdamOptimizer(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[43],cmap='Greys')

#plt.imshow(x\_test[321],cmap='Greys')

#plt.imshow(x\_test[495],cmap='Greys')

plt.imshow(x\_test[6065],cmap='Greys')

﻿**﻿categorical\_accuracy: 0.9954 - val\_loss: 0.1025 - val\_categorical\_accuracy: 0.9771**

****

** ﻿0.996665 /2**

** ﻿0.999836 / 7**

** ﻿0.652295 /2**

** ﻿0.997883 / 3**

**TEHTÄVÄ 12**

﻿import tensorflow as tf

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

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

#%%

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

#%%

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

y\_test=np.array(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(100, activation='relu'),

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

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

])

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

optimizers=tf.compat.v1.train.AdamOptimizer(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[43],cmap='Greys')

#plt.imshow(x\_test[321],cmap='Greys')

#plt.imshow(x\_test[231],cmap='Greys')

plt.imshow(x\_test[321],cmap='Greys')

#%%

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

#%%

model.save('mnistconvmodel.h5')

**Edellisenä tehtävän accuracy**

﻿**﻿categorical\_accuracy: 0.9954 - val\_loss: 0.1025 - val\_categorical\_accuracy: 0.9771**

**Tämän tehtävän accuracy**

**﻿categorical\_accuracy: 0.9956 - val\_loss: 0.0454 –**

**val\_categorical\_accuracy: 0.9895**

** 1 /3  ﻿0.460359 /2**

**TEHTÄVÄ 14**

﻿import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

df=pd.read\_csv("/Users/kayttaja/.spyder-py3/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

#%%

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 (k$)')

ax.set\_zlabel('Spending Score (1-100)')

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("/Users/kayttaja/.spyder-py3/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

#%%

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

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(0,4):

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 (k$)')

ax.set\_zlabel('Spending Score (1-100)')

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

seqlenght=12

df=pd.read\_csv("/Users/kayttaja/.spyder-py3/monthly-car-sales.csv")

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,seqlenght):

df['SalesDiffLag'+str(i)]=df['SalesDiff'].shift(1)

for i in range(1,ennusteaika+1):

df['SalesDiffFut'+str(i)]=df['SalesDiff'].shift(1)

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

df\_train.dropna(inplace=True)

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

#%%

input\_vars=['SalesDiff']

for i in range(1,seqlenght):

input\_vars.append('SalesDiffLag'+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],seqlenght,1)

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],seqlenght,1)

#%%

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\_

#%%

modelLSTM = tf.keras.Sequential([

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

return\_sequences=False),

tf.keras.layers.Dense(ennusteaika)

])

modelLSTM.compile(loss="mse",

optimizers=tf.compat.v1.train.AdamOptimizer(learning\_rate=0.001),

metrics =['mae'])

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

#%%

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=np.array(ennuste[1:])