```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np
(x_train, y_train), (x_test, y_test) =
keras.datasets.fashion_mnist.load_data()
plt.imshow(x_train[1])
plt.imshow(x_train[0])
x_train = x_train.astype('float32') / 255.0 x_test =
x_test.astype('float32') / 255.0 x_train =
x_train.reshape(-1, 28, 28, 1) x_test = x_test.reshape(-1,
28, 28, 1)
x_train.shape (60000, 28, 28) x_test.shape (10000, 28, 28, 1)
y_train.shape (60000,)
y_test.shape (10000,)
: model =
keras.Sequential([
keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
import numpy as np from tensorflow.keras.models
import Sequential from tensorflow.keras.layers import
Dense, Dropout from tensorflow.keras.optimizers
import RMSprop from tensorflow.keras.datasets
import mnist import matplotlib.pyplot as plt from
sklearn import metrics
import pandas as pd
data=pd.read_csv("adress")
(x_train, y_train), (x_test, y_test) = mnist.load_data()
plt.imshow(x_train[0],
cmap='gray')
plt.show()
print(x_train[0])
print("X_train shape", x_train.shape)
print("y_train shape", y_train.shape)
print("X_test shape", x_test.shape) print("y_test
shape", y_test.shape) X_train shape (60000,
28, 28) y_train shape (60000,)
X_test shape (10000, 28, 28) y_test shape
(10000,)
validation_data=(x_test, y_test))
set and validation set yourself,
# 60000image/128=469 batch each
# Evaluate the model
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0]) print('Test accuracy:', score[1])
```

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
import numpy as np
(x_train, y_train), (x_test, y_test) =
keras.datasets.fashion_mnist.load_data()
plt.imshow(x_train[1])
plt.imshow(x_train[0])
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
x_train.shape
x_test.shape
y_train.shape
y_test.shape
model = keras.Sequential([
keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1))
keras.layers.MaxPooling2D((2,2)),
keras.layers.Dropout(0.25),
keras.layers.Conv2D(64, (3,3), activation='relu')
keras.layers.MaxPooling2D((2,2)),
keras.layers.Dropout(0.25),
keras.layers.Conv2D(128, (3,3), activation='relu'),
keras.layers.Flatten(),
keras.layers.Dense(128, activation='relu'),
keras.layers.Dropout(0.25),
keras.layers.Dense(10, activation='softmax')])
model.summary()
Model: "sequential"
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test,
y_test))
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
boston = load_boston()
data = pd.DataFrame(boston.data)
data.head()
data.columns = boston.feature_names
data['PRICE'] = boston.target
data.head(n=10)
print(data.shape)
data.isnull().sum()
data.describe()
data.info()
import seaborn as sns
sns.distplot(data.PRICE)
sns.boxplot(data.PRICE)
correlation = data.corr()
correlation.loc['PRICE']
import matplotlib.pyplot as plt
fig,axes = plt.subplots(figsize=(15,12))
sns.heatmap(correlation,square = True,annot = True)
plt.figure(figsize = (20,5))
features = ['LSTAT','RM','PTRATIO']
for i, col in enumerate(features):
plt.subplot(1, len(features) , i+1)
x = data[col]
y = data.PRICE
plt.scatter(x, y, marker='o')
plt.title("Variation in House prices")
plt.xlabel(col)
plt.ylabel('"House prices in $1000"')
X = data.iloc[:,:-1]
y= data.PRICE
mean = X_train.mean(axis=0)
std = X_train.std(axis=0)
X_train = (X_train - mean) / std
X_test = (X_test - mean) / std
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred = regressor.predict(X_test)
from sklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
print(rmse)
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print(r2)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
import keras
from keras.layers import Dense, Activation,Dropout
from keras.models import Sequential
```

```
model = Sequential()
model.add(Dense(128,activation = 'relu',input_dim =13))
model.add(Dense(64,activation = 'relu'))
model.add(Dense(32,activation = 'relu'))
model.add(Dense(16,activation = 'relu'))
model.add(Dense(1))
model.compile(optimizer = 'adam',loss
='mean_squared_error',metrics=['mae'])
!pip install ann_visualizer
!pip install graphviz
from ann_visualizer.visualize import ann_viz;
ann_viz(model, title="DEMO ANN");
history = model.fit(X_train, y_train, epochs=100, validation_split=0.05)
from plotly.subplots import make_subplots
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Scattergl(y=history.history['loss'],
name='Train'))
fig.add_trace(go.Scattergl(y=history.history['val_loss'],
name='Valid'))
fig.update_layout(height=500, width=700,
xaxis_title='Epoch',
yaxis_title='Loss')
fig.show()
fig = go.Figure()
fig.add_trace(go.Scattergl(y=history.history['mae'],
name='Train'))
fig.add_trace(go.Scattergl(y=history.history['val_mae'],
name='Valid'))
fig.update_layout(height=500, width=700,
xaxis_title='Epoch',
yaxis_title='Mean Absolute Error')
fig.show()
y_pred = model.predict(X_test)
mse_nn, mae_nn = model.evaluate(X_test, y_test)
print('Mean squared error on test data: ', mse_nn)
print('Mean absolute error on test data: ', mae_nn)
from sklearn.metrics import mean_absolute_error
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
mae_lr = mean_absolute_error(y_test, y_pred_lr)
print('Mean squared error on test data: ', mse_lr)
print('Mean absolute error on test data: ', mae_lr)
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)
print(r2)
from sklearn.metrics import mean_squared_error
rmse = (np.sqrt(mean_squared_error(y_test, y_pred)))
print(rmse)
import sklearn
new_data = sklearn.preprocessing.StandardScaler().fit_transform(([[0.1,
10.0,
5.0, 0, 0.4, 6.0, 50, 6.0, 1, 400, 20, 300, 10]]))
prediction = model.predict(new_data)
print("Predicted house price:", prediction)
```