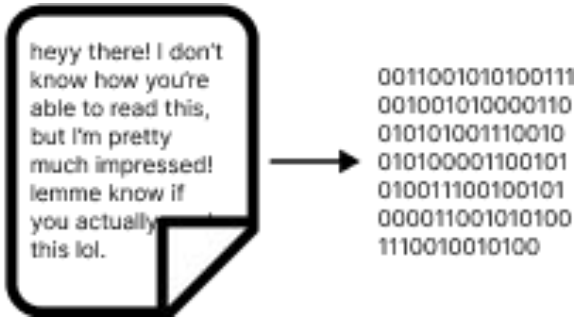
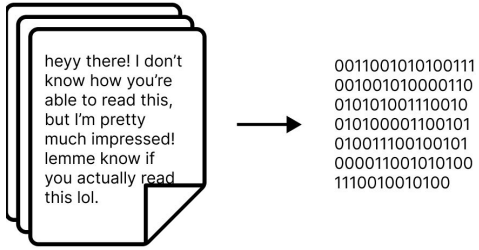


the art of **data pre-processing**

improving the performance of deep learning models

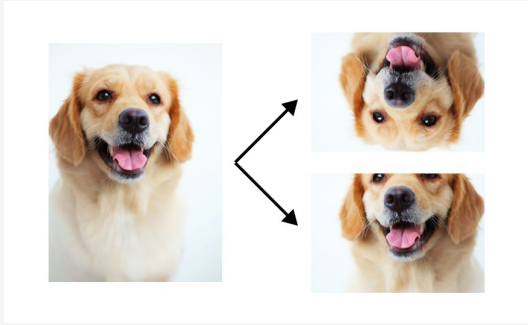
absolute necessity.

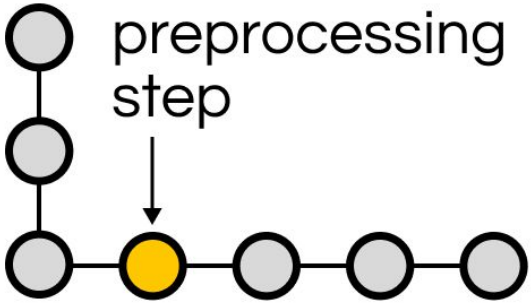
sometimes data preprocessing is necessary before moving on with building our machine learning model.



good-to-have.

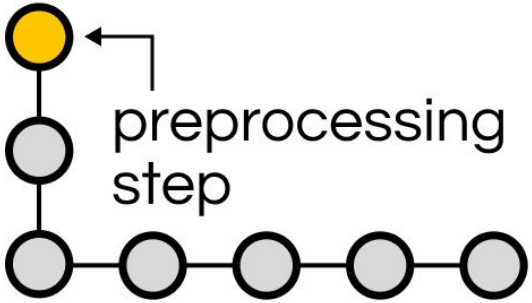
and sometimes data preprocessing is not necessary but it's good to have because it boosts performance.





01 inside the model

preprocessing is done inside a pipeline to reduce the **training - serving skew**.



02 outside the model

preprocessing is done outside a pipeline for **better understanding** of the data.

keras preprocessing layers api

Keras preprocessing layers are composable and customizable building blocks that cover all common steps of the data preprocessing workflow.

keras.layers.TextVectorization
keras.layers.CategoryEncoding
keras.layers.Rescaling
etc...

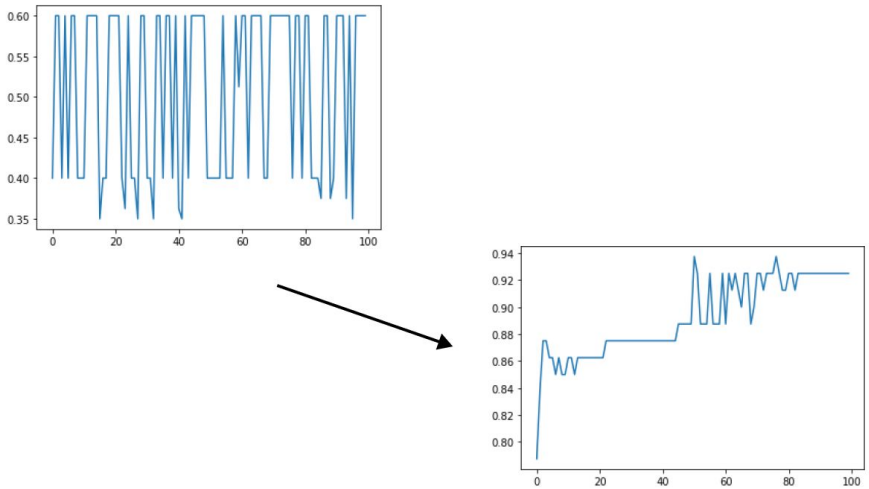
practical example #1

Structured data

Graphical Data

keras.layers.Normalization

keras.layers.StringLookup



```python
# General purpose
import numpy as np
import pandas as pd
from sklearn.model_selection import
train_test_split

# For data visualization
import seaborn as sns
import matplotlib.pyplot as plt

# For building neural network using Keras
import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense
```

```python
df = pd.read_csv("/content/Social_Network_Ads.csv")
df
```

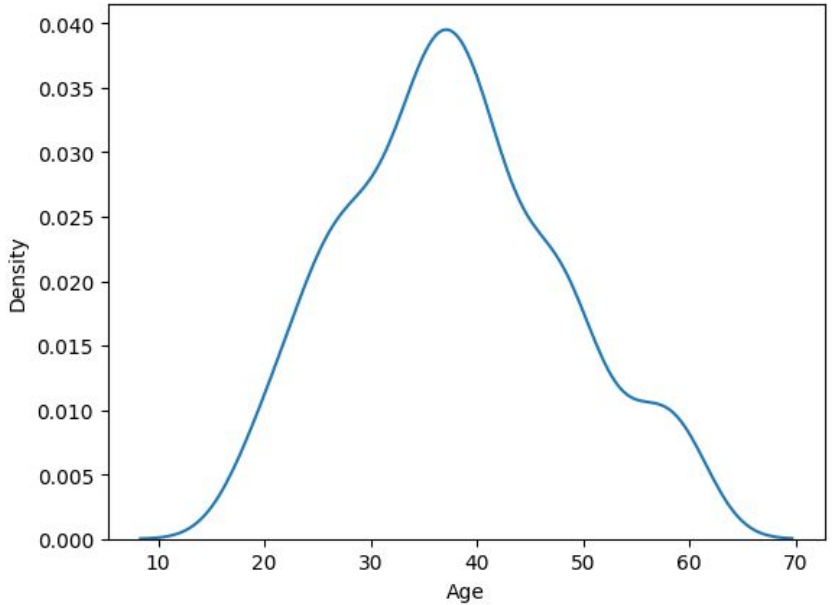
	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

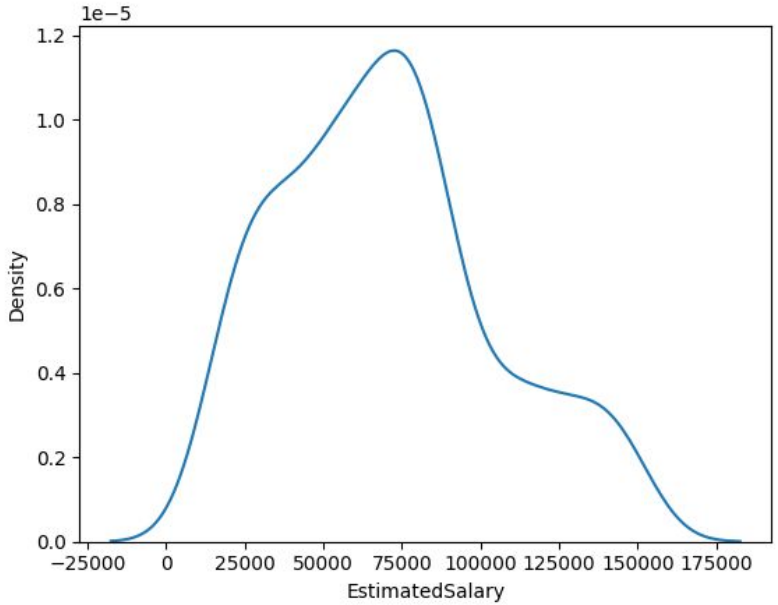
400 rows × 5 columns

```
data = df.iloc[:, 2:]
data
```

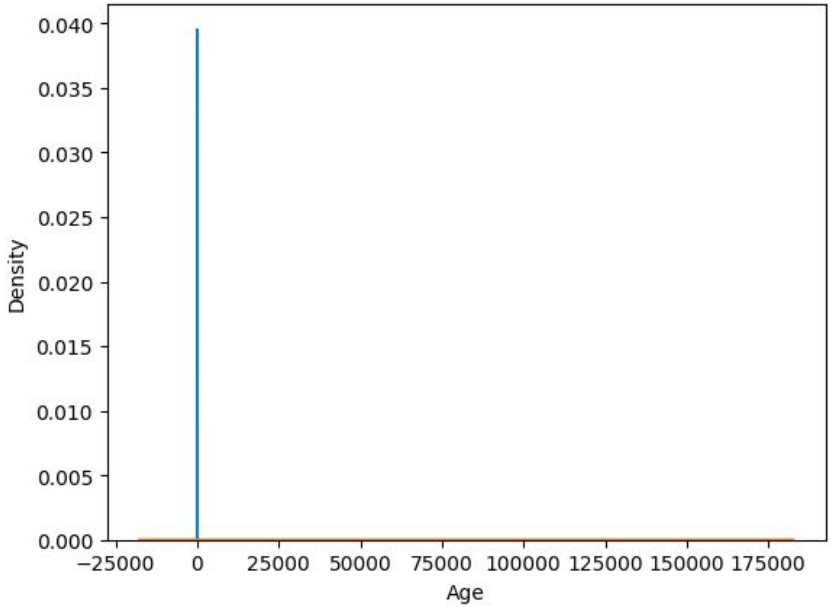
	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
...	...	...	...
395	46	41000	1
396	51	23000	1
397	50	20000	1
398	36	33000	0
399	49	36000	1

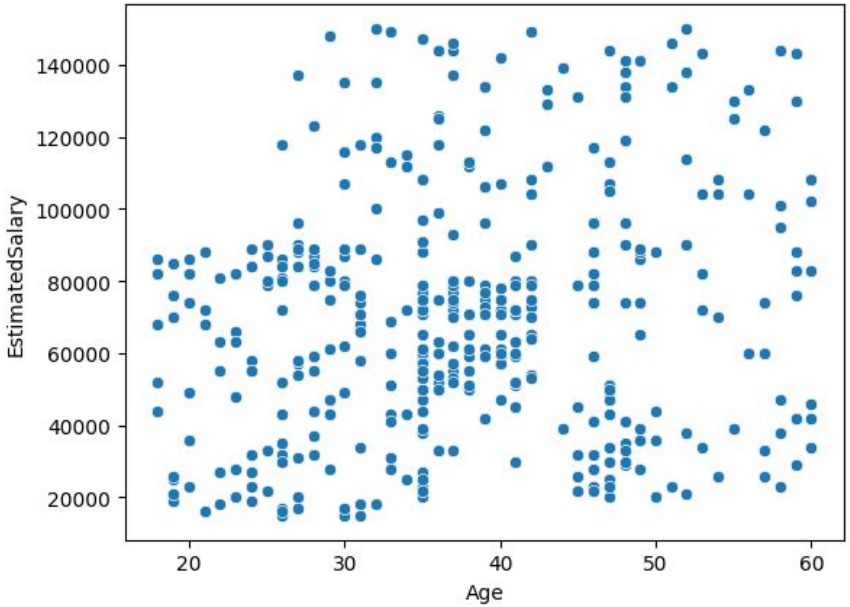
400 rows × 3 columns





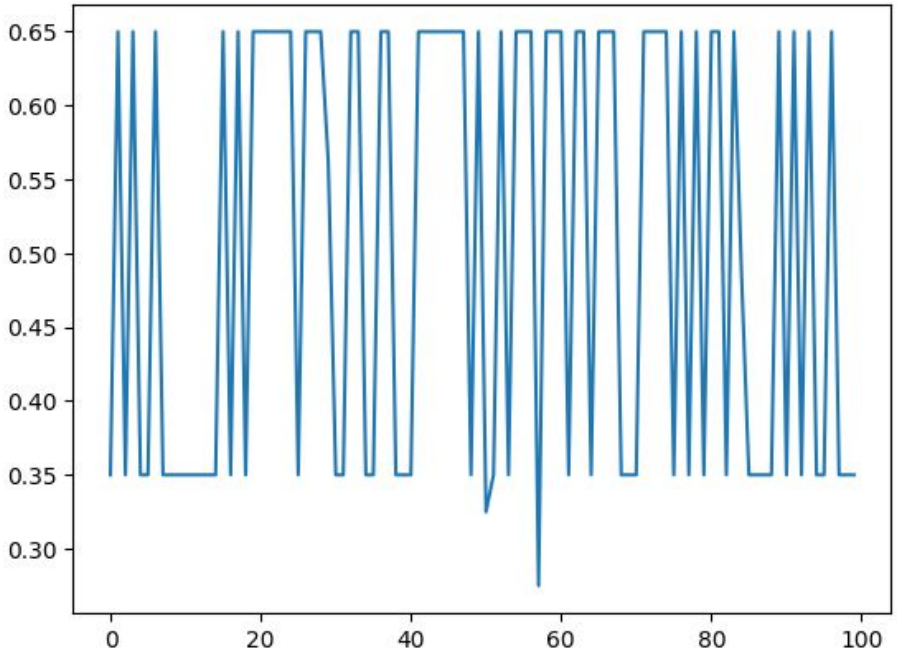
seems quite normal?


0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
−25000
0
25000
50000
75000
100000
125000
150000
175000
Age


140000
120000
100000
80000
60000
40000
20000
EstimatedSalary
20
30
40
50
60
Age

```
X = data.iloc[:, 0:2]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.2, random_state=42)
```

```
model = Sequential()

model.add(Dense(128,activation='relu',input_dim=2))
model.add(Dense(1,activation='sigmoid'))

model.compile(optimizer='adam',loss='binary_crossentropy',
metrics=['accuracy'])

history = model.fit(X_train,y_train,validation_data=
(X_test,y_test),epochs=100)
```



	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
...	...	...	...	...
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

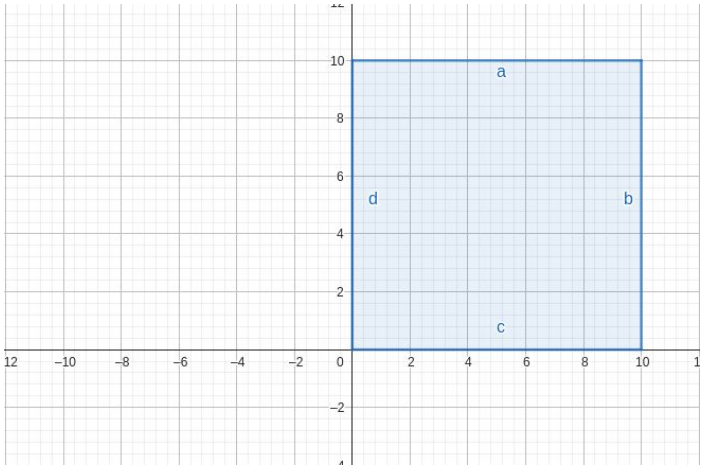
400 rows × 4 columns

```
from keras.layers import StringLookup

layer = StringLookup(vocabulary=['Male',
'Female'], output_mode='one_hot')

gender_ohe = layer(data2['Gender']).numpy()

gender_df = pd.DataFrame({'gender_male':
gender_ohe[:,1], 'gender_female':
gender_ohe[:,2]})

data2.drop(['Gender'], axis=1, inplace=True)
data2 = pd.concat([data2, gender_df],
axis=1)
data2
```

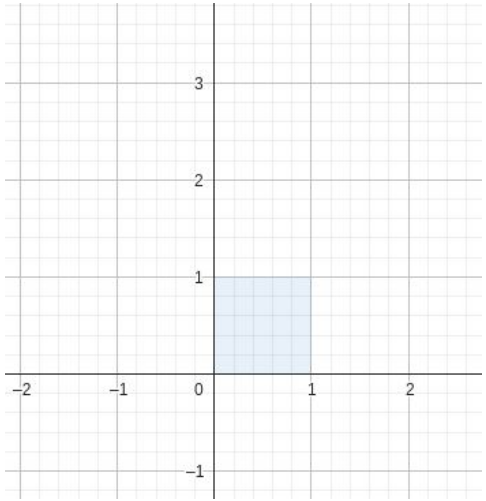
	Age	EstimatedSalary	Purchased	gender_male	gender_female
0	19	19000	0	1.0	0.0
1	35	20000	0	1.0	0.0
2	26	43000	0	0.0	1.0
3	27	57000	0	0.0	1.0
4	19	76000	0	1.0	0.0
...	...	...	...	...	...
395	46	41000	1	0.0	1.0
396	51	23000	1	1.0	0.0
397	50	20000	1	0.0	1.0
398	36	33000	0	1.0	0.0
399	49	36000	1	0.0	1.0

400 rows × 5 columns

What is **normalization**?



original data



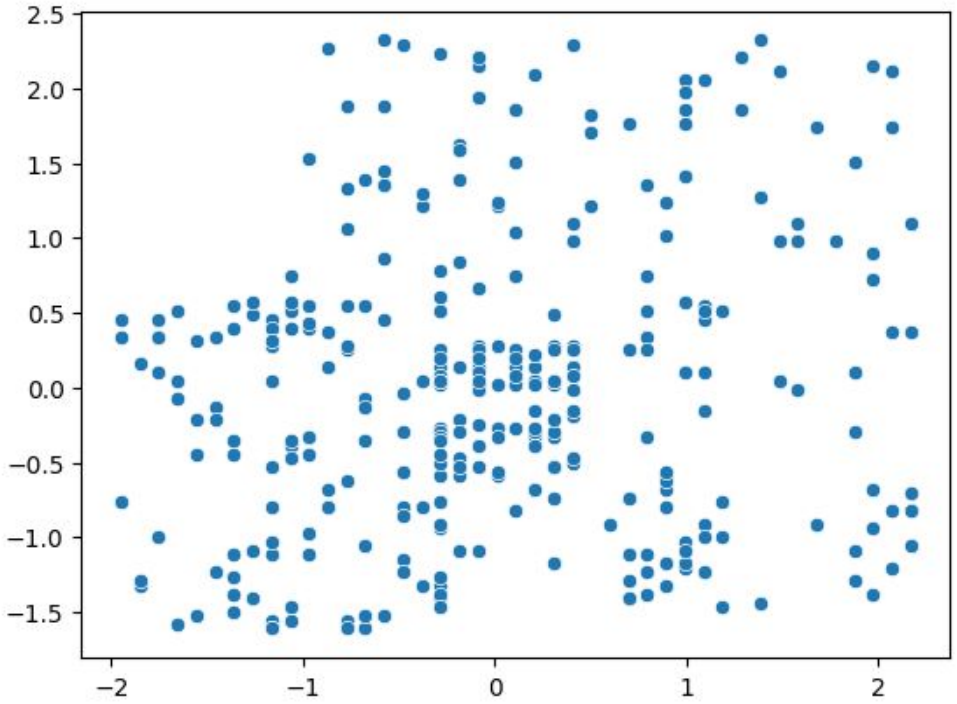
scaled data

min max scaling: $$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

```python
X = data2.drop('Purchased', axis=1)
y = data2['Purchased']

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=.2, random_state=42)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

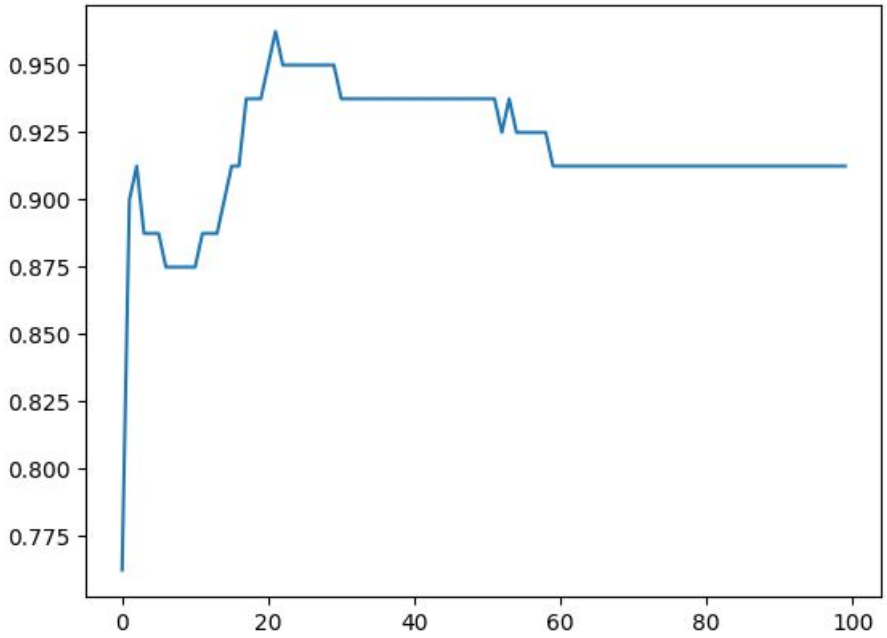
```python
# If done in Keras
adapt_data = np.array([1., 2., 3., 4., 5.],
dtype='float32')

input_data = np.array([1., 2., 3.], dtype='float32')

layer = tf.keras.layers.Normalization(axis=None)
layer.adapt(adapt_data)
layer(input_data)
# <tf.Tensor: shape=(3,), dtype=float32, numpy=
# array([-1.4142135, -0.70710677, 0.], dtype=float32)
```

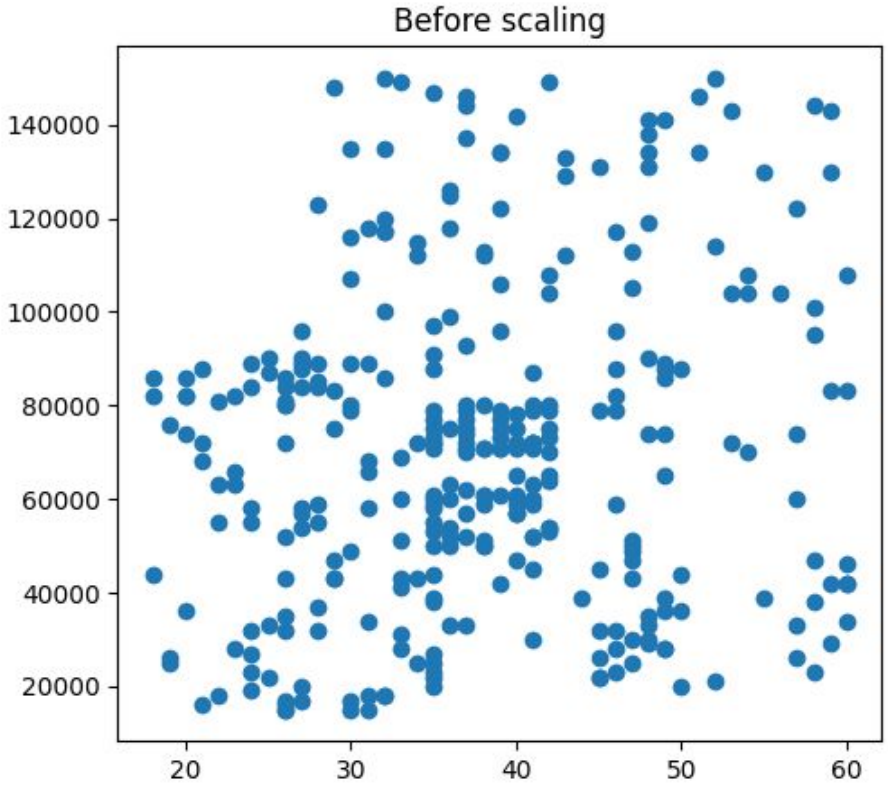
```python
sns.scatterplot(x=X_train_scaled[:,0],
y=X_train_scaled[:, 1])
```

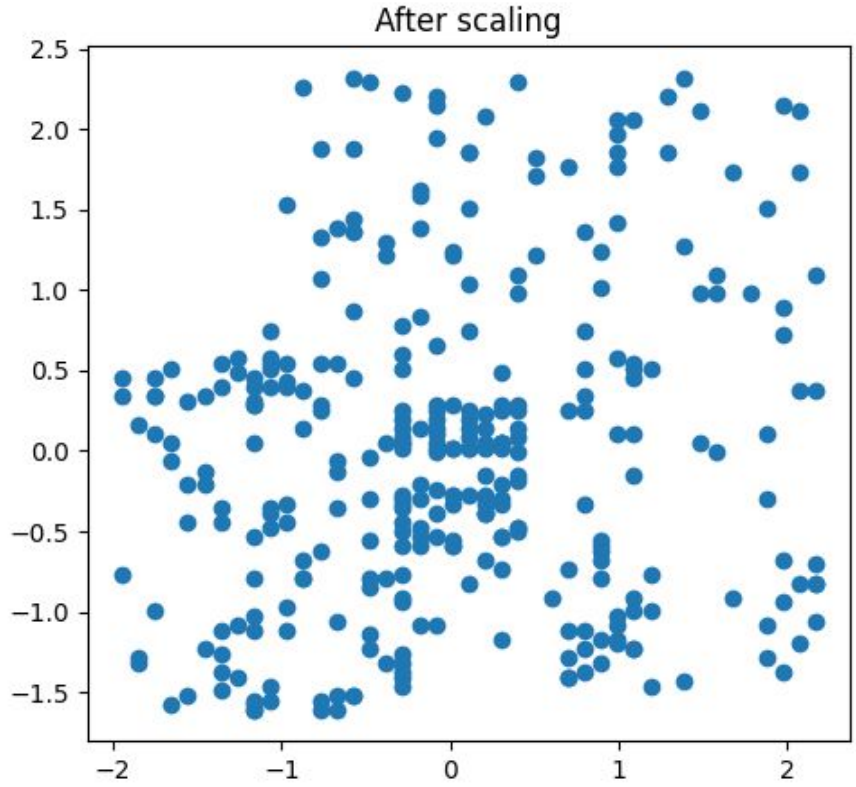


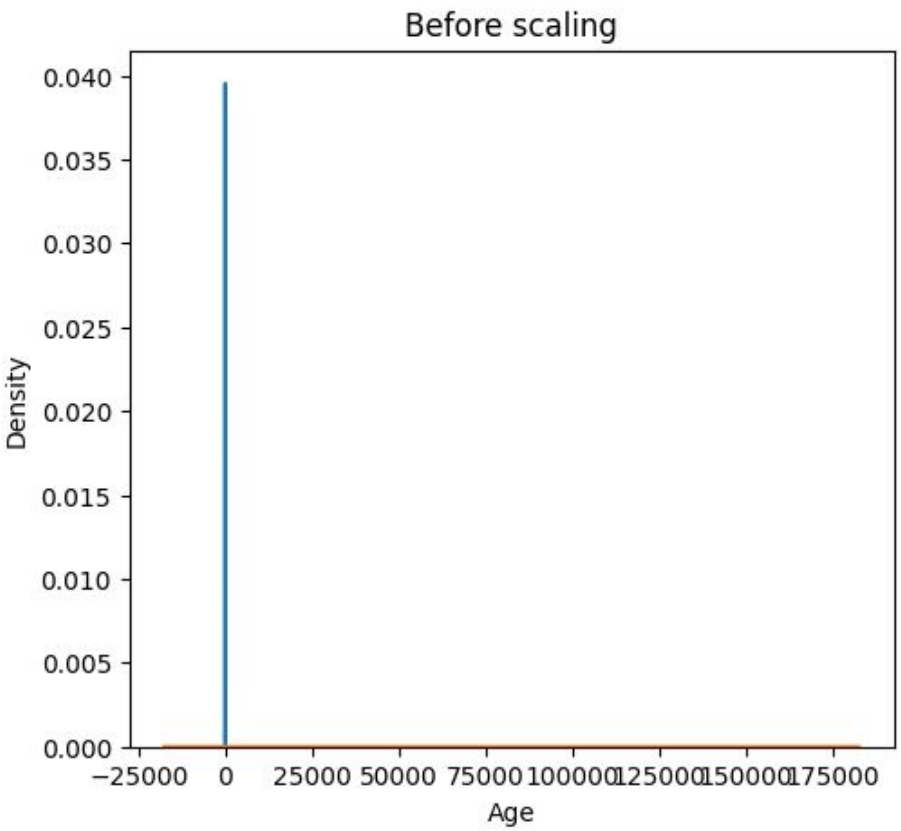
```
model2 = Sequential()

model2.add(Dense(128,activation='relu',input_dim=4))
model2.add(Dense(1,activation='sigmoid'))

model2.compile(optimizer='adam',loss='binary_crossentropy',
metrics=['accuracy'])

history2 =
model2.fit(X_train_scaled,y_train,validation_data=
(X_test_scaled,y_test),epochs=100)
```

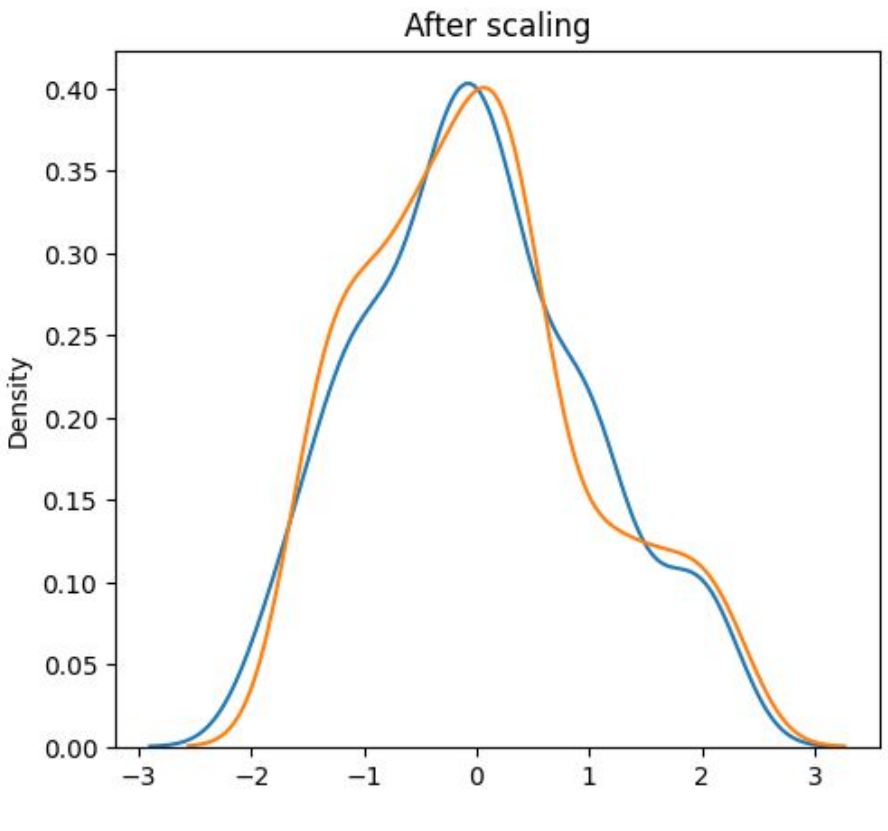
```
plt.plot(history2.history['val_accuracy'])
```

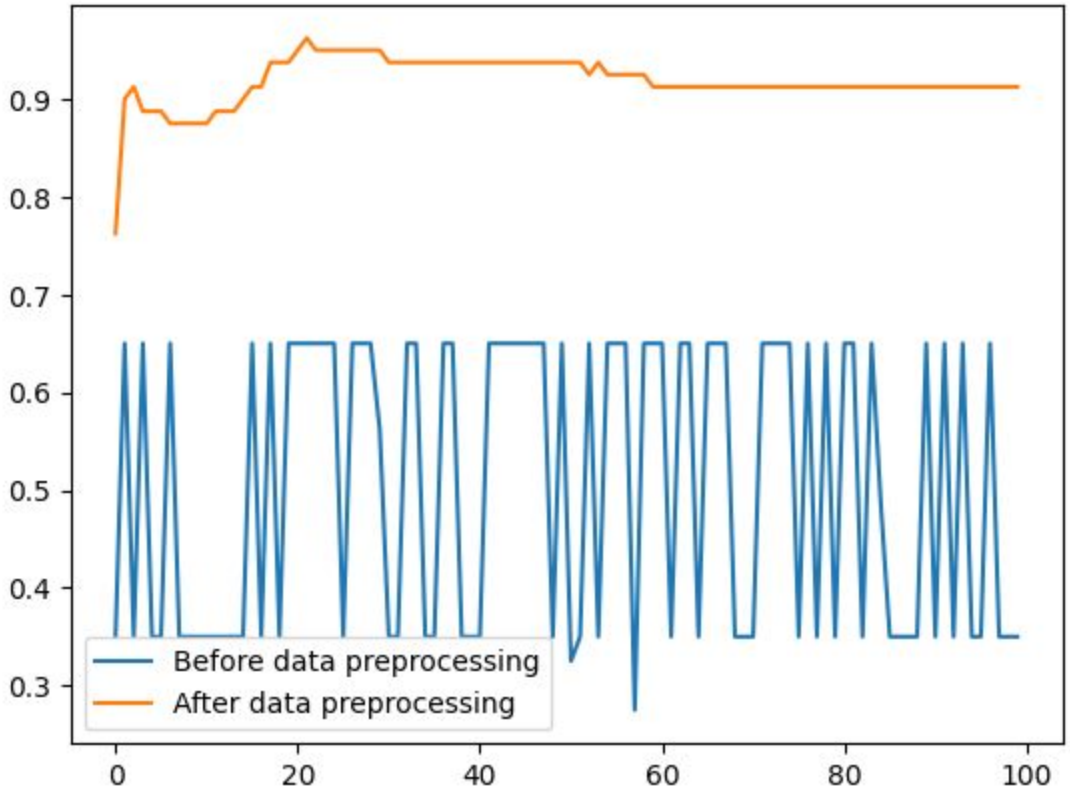



Before scaling
140000
120000
100000
80000
60000
40000
20000
20
30
40
50
60


After scaling
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2
−1
0
1
2


Before scaling
Density
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−25000
0
25000
50000
75000
100000
125000
150000
175000
Age


After scaling
Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−3
−2
−1
0
1
2
3


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
Before data preprocessing
After data preprocessing

practical example #2

Structured data

Graphical Data

keras.layers.Rescaling

keras.layers.BatchNormalization

keras.preprocessing.image.ImageDataGenerator

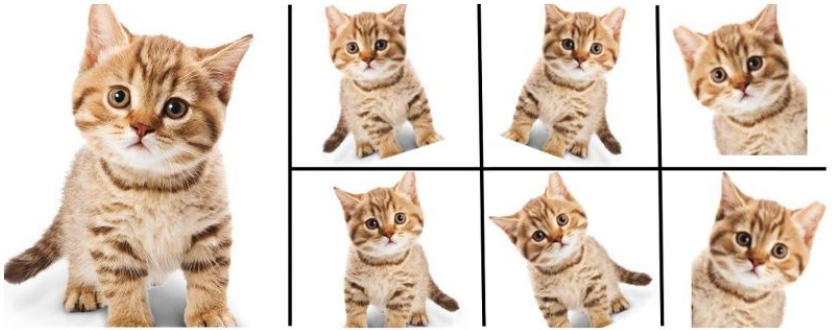


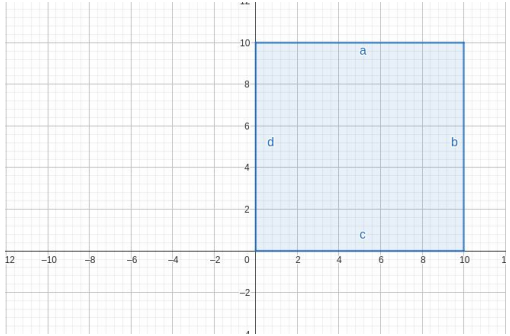
Image classification

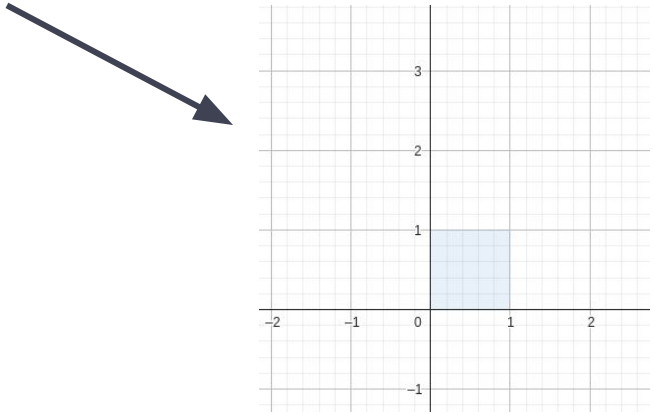
Dogs - 1000 Samples

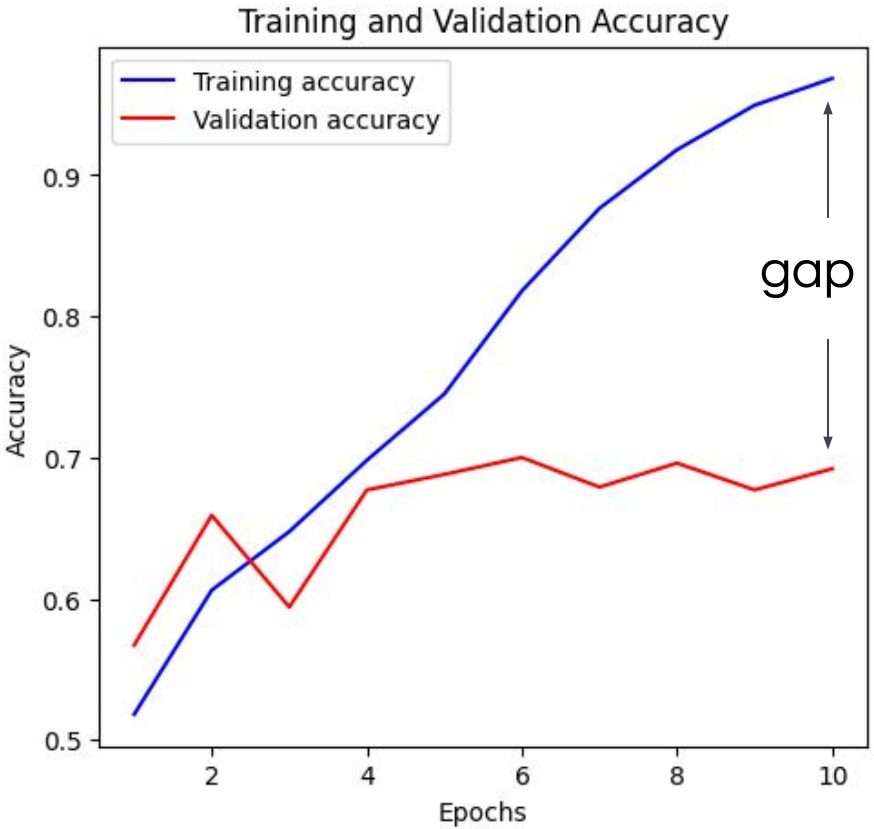
Cats - 1000 Samples

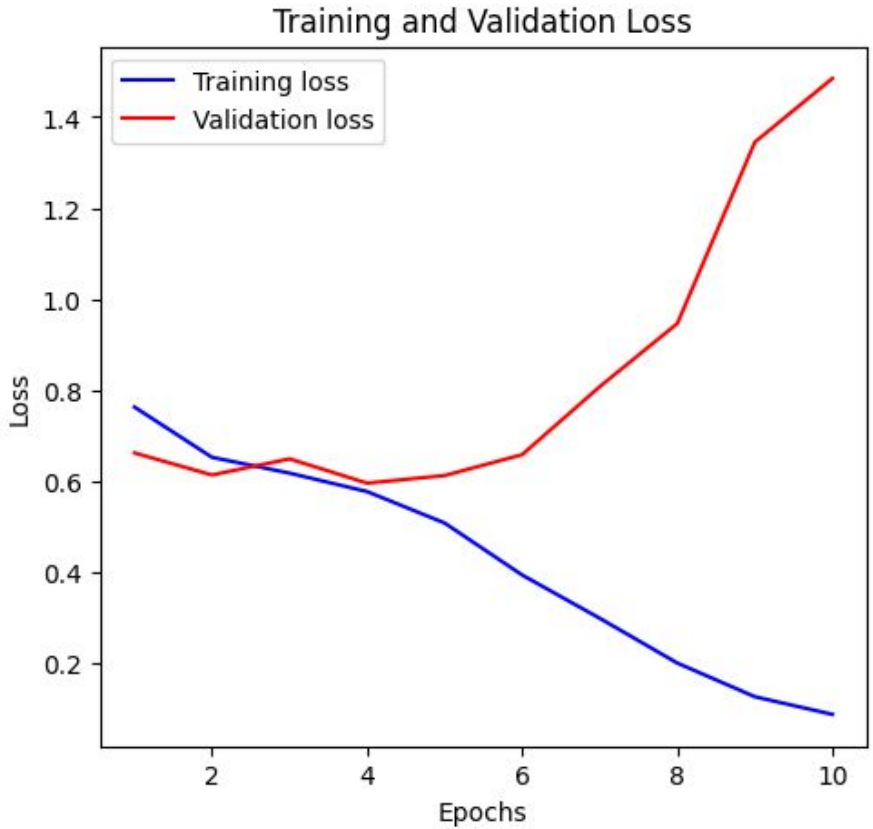
```python
model = Sequential()

# Scaling layer
model.add(Rescaling(scale = 1./255, input_shape=(256, 256, 3)))

# Rest of the code is normal model building

model.add(Conv2D(32, kernel_size=(3, 3), padding='valid',
activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding='valid'))

# ...some more layers

model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
```

```python
tf.keras.layers.Rescaling(scale, offset=0.0, **kwargs)
```

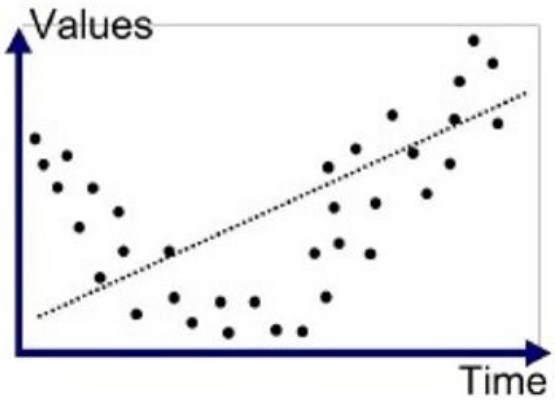




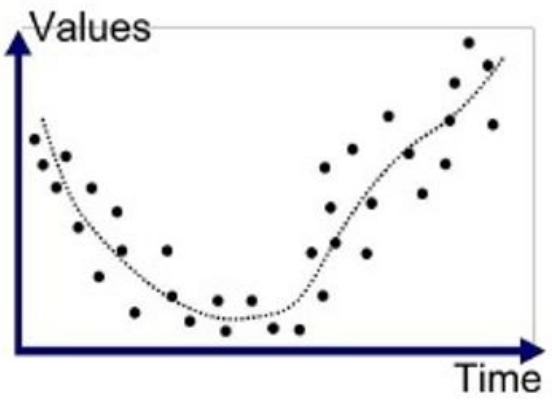

Training and Validation Accuracy
Training accuracy
Validation accuracy
gap
Accuracy
Epochs
0.5
0.6
0.7
0.8
0.9
2
4
6
8
10


Training and Validation Loss
Training loss
Validation loss
Loss
Epochs
0.2
0.4
0.6
0.8
1.0
1.2
1.4
2
4
6
8
10

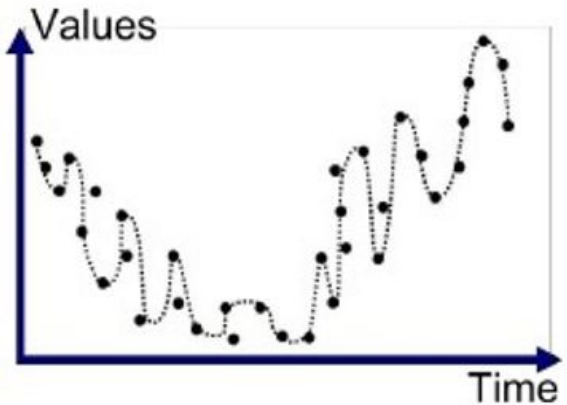
overfitting?



Underfitted



Good Fit/Robust



Overfitted


A


Competitive | Inferno
0:07
CS:GO
HS% K/D ADR UD EF Score
5
team_ZqZer
Alive: 0/5
39 ZqZer 53 1.08 94 9 3 29
85 -mFK 66 0.92 90 157 2 26
70 Akatsuki AKANJID 50 0.83 68 14 0 22
41 Enemy Purple 44 0.81 76 22 9 21
22 Enemy Blue 14 0.50 51 13 8 16
5 0
1st 2nd
10 0
10
team_shrekuss
Alive: 2/5
59 гений DAEMON228 66 0.90 76 0 4 37
31 mouz innersh1ne 41 1.54 124 0 6 36
50 ★ Global VoxDaKing 46 1.44 84 46 2 36
55 shrekuss 42 1.27 82 0 0 29
82 pro100kvasha 33 0.90 58 19 3 22
SERVER WEBSITE
[MOUSE2] Enable Cursor


Competitive | Inferno
0:09
CS:GO
Money K A D MVP Score
15
team_shrekuss
Alive: 5/5
34 mouz innersh1ne $4650 26 4 15 59
50 ★ Global VoxDaKing $0 23 5 14 58
60 гений DAEMON228 $4200 15 4 14 50
57 shrekuss $4000 16 0 16 33
92 pro100kvasha $1950 10 1 12 25
10 5
1st 2nd
5 2
Loss Bonus
7
team_ZqZer
Alive: 5/5
86 -mFK 18 3 19 40
71 Akatsuki AKANJID 16 2 18 38
43 ZqZer 15 4 18 36
43 Enemy Purple 14 3 16 31
28 Enemy Blue 7 2 19 16
SERVER WEBSITE
[MOUSE2] Enable Cursor

but how do i prevent
overfitting?

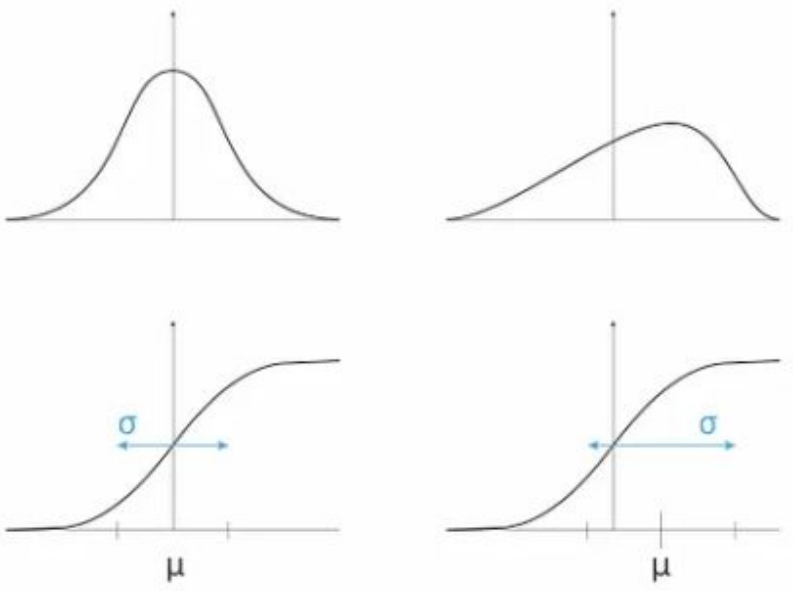
Batch Normalization

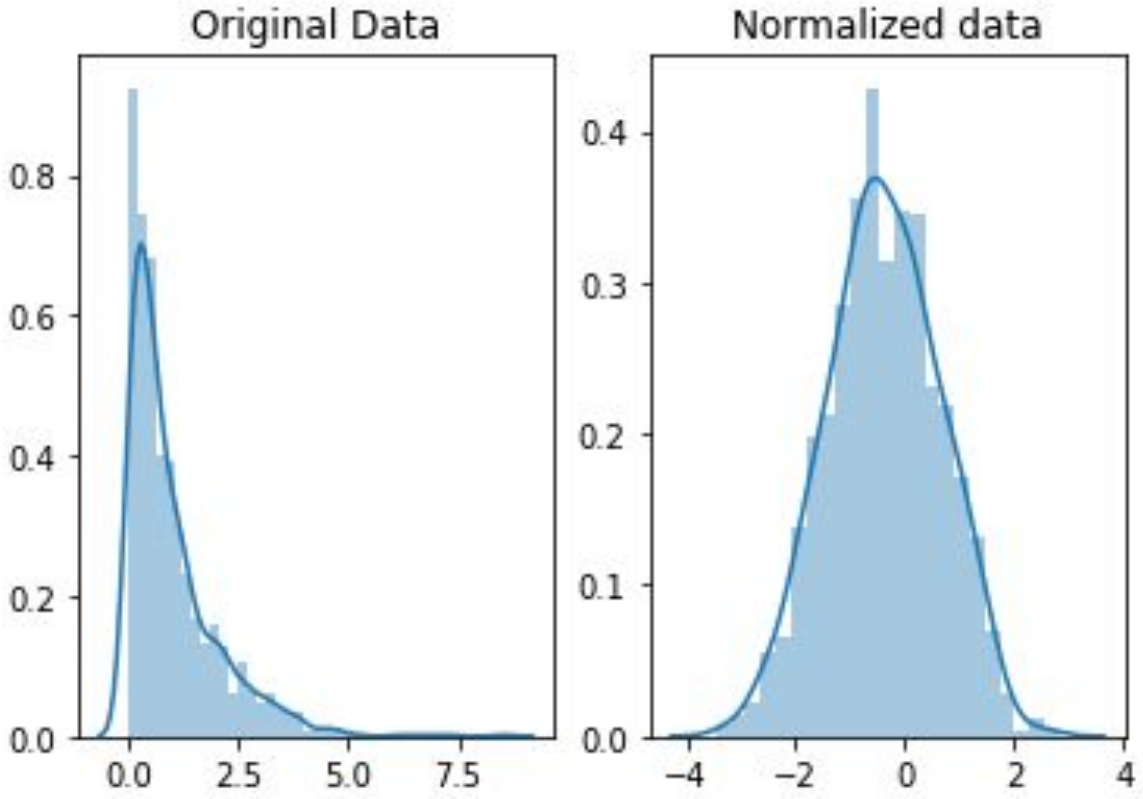
(1) $\mu = \frac{1}{n}\sum_i Z^{(i)}$

(2) $\sigma^2 = \frac{1}{n}\sum_i (Z^{(i)} - \mu)^2$

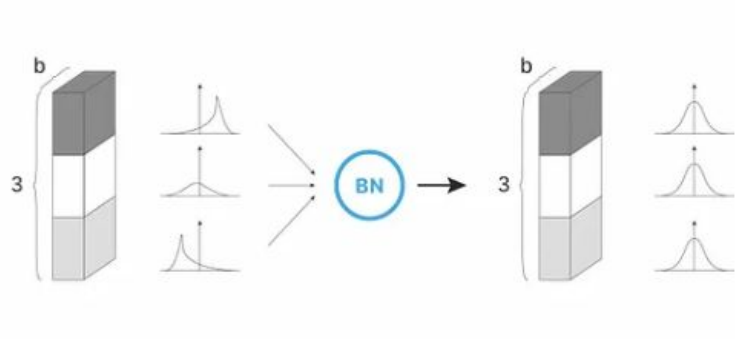
(3) $Z^{(i)}_{norm} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}}$

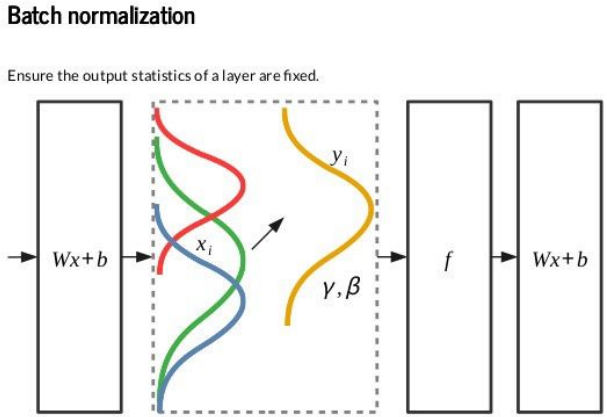
(4) $\breve{Z} = \gamma * Z^{(i)}_{norm} + \beta$





keras.layers.BatchNormalization






```
model2.add(Conv2D(32, kernel_size=(3, 3),
padding='valid', activation='relu'))
model2.add(BatchNormalization()) # Normalization
model2.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding='valid'))

model2.add(Conv2D(64, kernel_size=(3, 3),
padding='valid', activation='relu'))
model2.add(BatchNormalization()) # Normalization
model2.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding='valid'))

model2.add(Conv2D(128, kernel_size=(3, 3),
padding='valid', activation='relu'))
model2.add(BatchNormalization()) # Normalization
model2.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding='valid'))
```

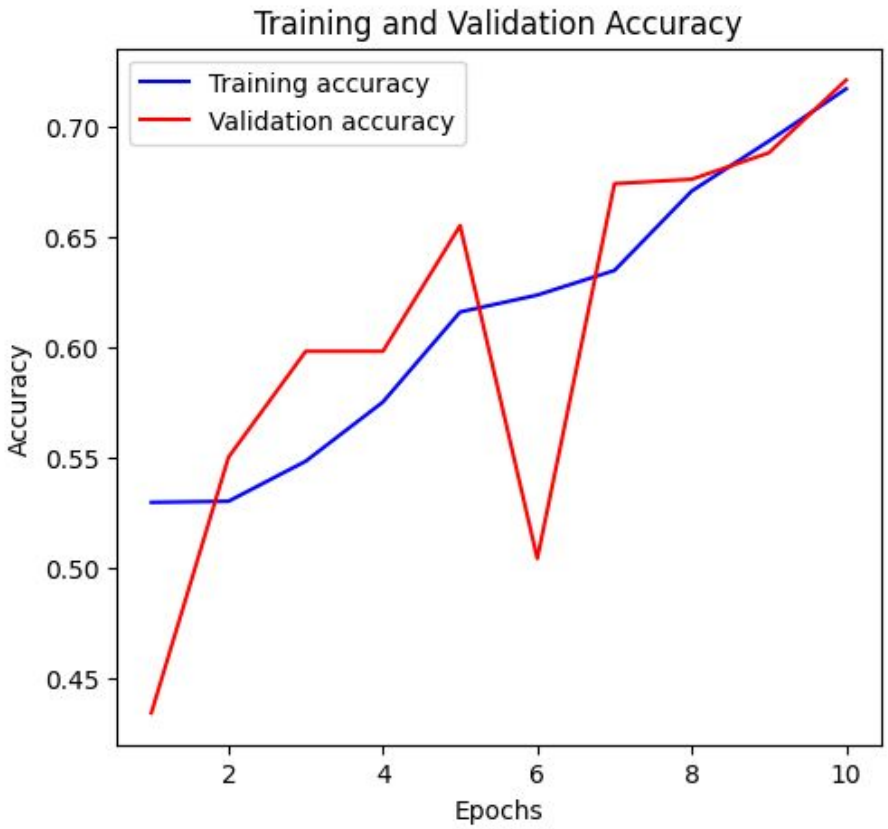
data augmentation

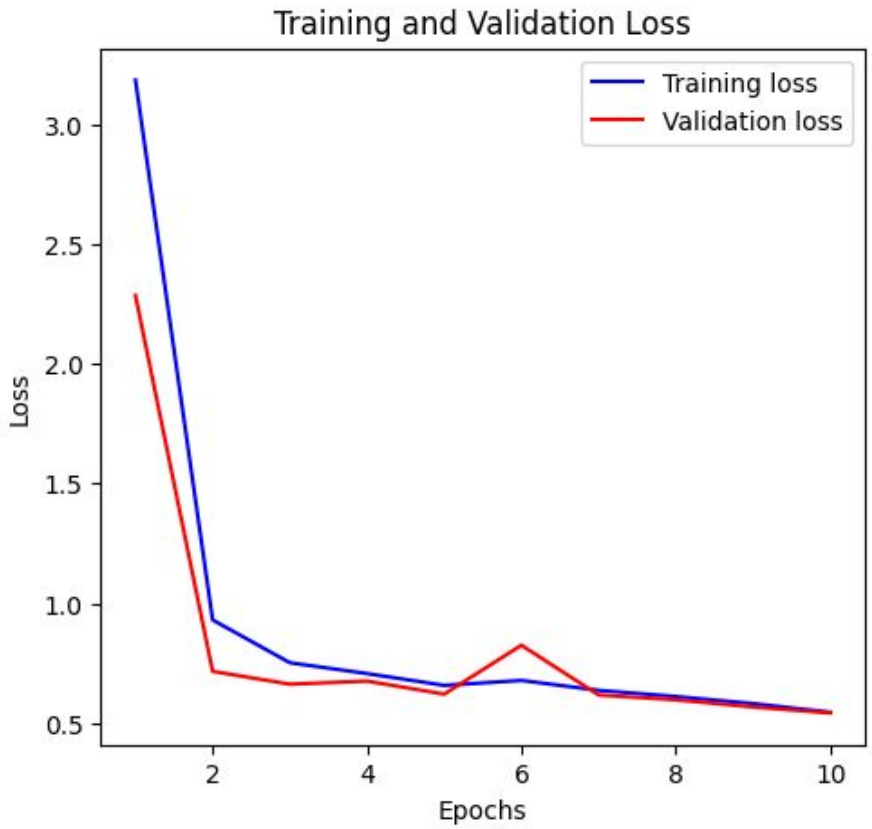
keras.preprocessing.image.ImageDataGenerator

keras.preprocessing.image.ImageDataGenerator

now we have 4000 + 4000 samples

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```


Training and Validation Accuracy
Training accuracy
Validation accuracy
Accuracy
0.70
0.65
0.60
0.55
0.50
0.45
2
4
6
8
10
Epochs


Training and Validation Loss
Training loss
Validation loss
Loss
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10
Epochs

that’s it for today!

- improve the model i showed
- continue learning how data preprocessing affects deep learning model
- explore the keras api
- enjoy rest of the sessions

thank you!

hoping to see y’all again