Knowledge Discovery and Data Mining

Lab 13 Using Keras 1 Regression Task

Xuan Song
Songx@sustech.edu.cn

Keras + TF

使用 Keras 心得

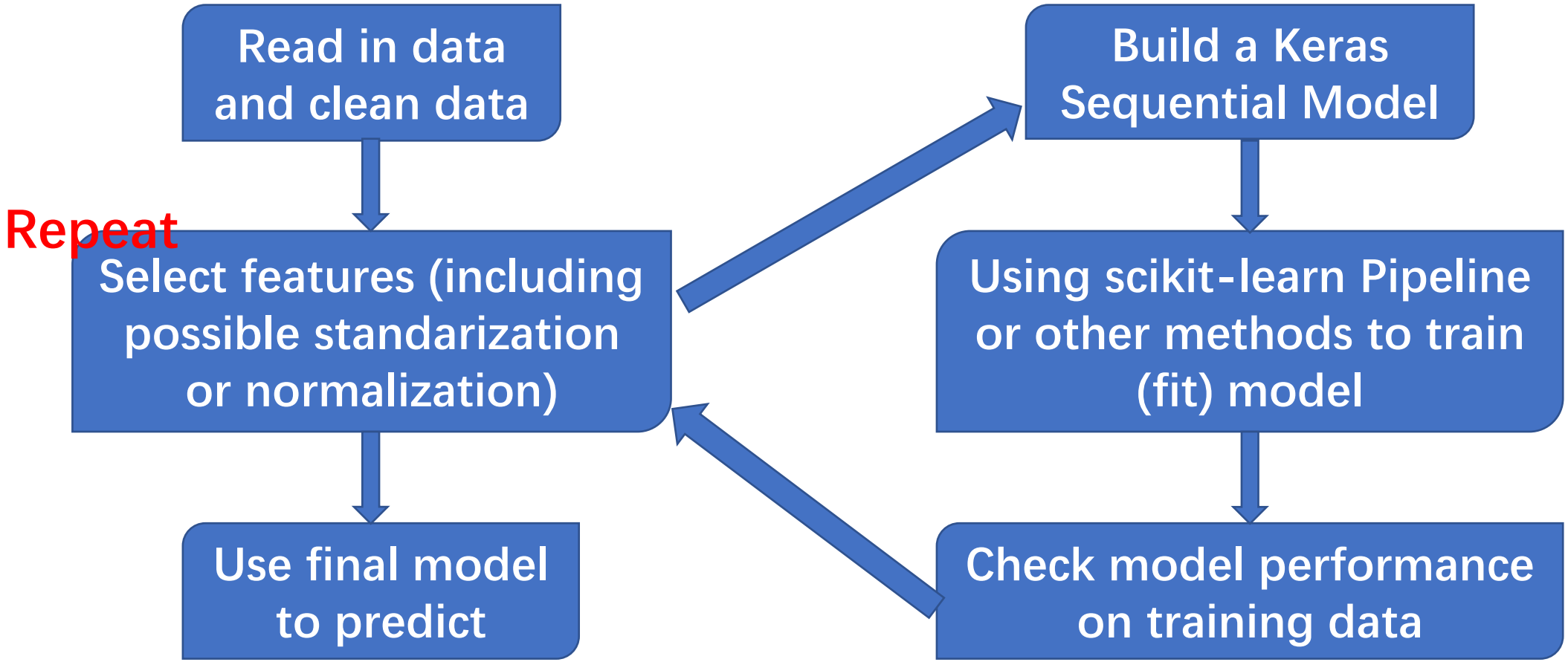
感謝 沈昇勳 同學提供圖檔

Install

We will be using these following libraries today:

- **tensorflow** (any version should be okay), keras is already intergrated into tf, so you don't have to install a keras separately.
- **seaborn**, for drawing plots, matplotlib is already intergrated into seaborn, so you don't have to install a matplotlib separately.
- **pandas**, really useful.
- **scikit-learn**, also very useful.

How -- General Guidence



Data we use

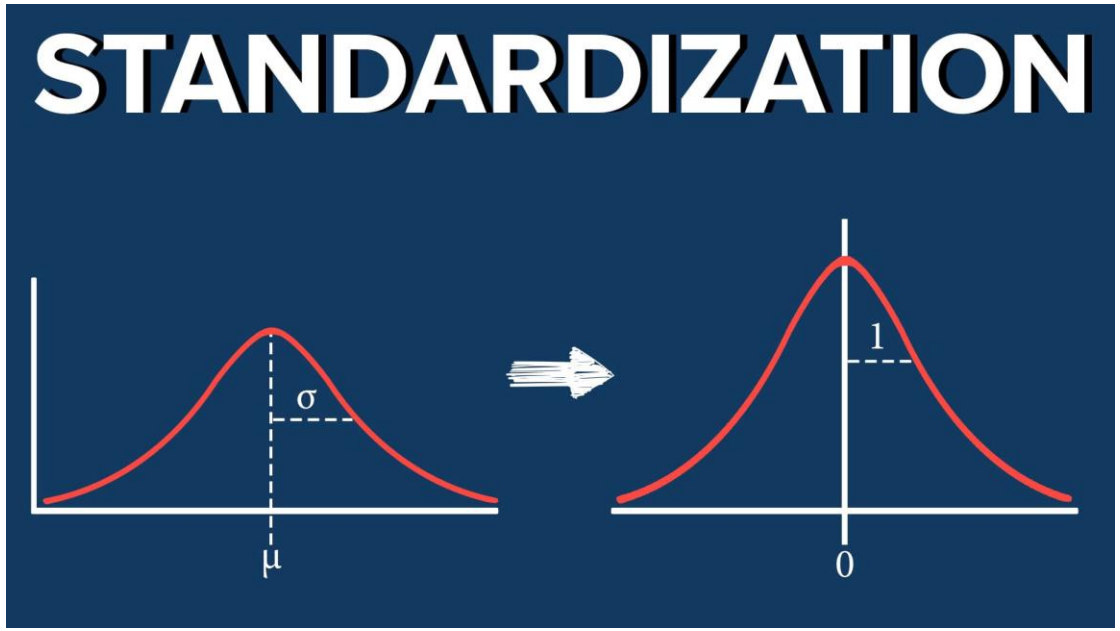
auto-mpg.data

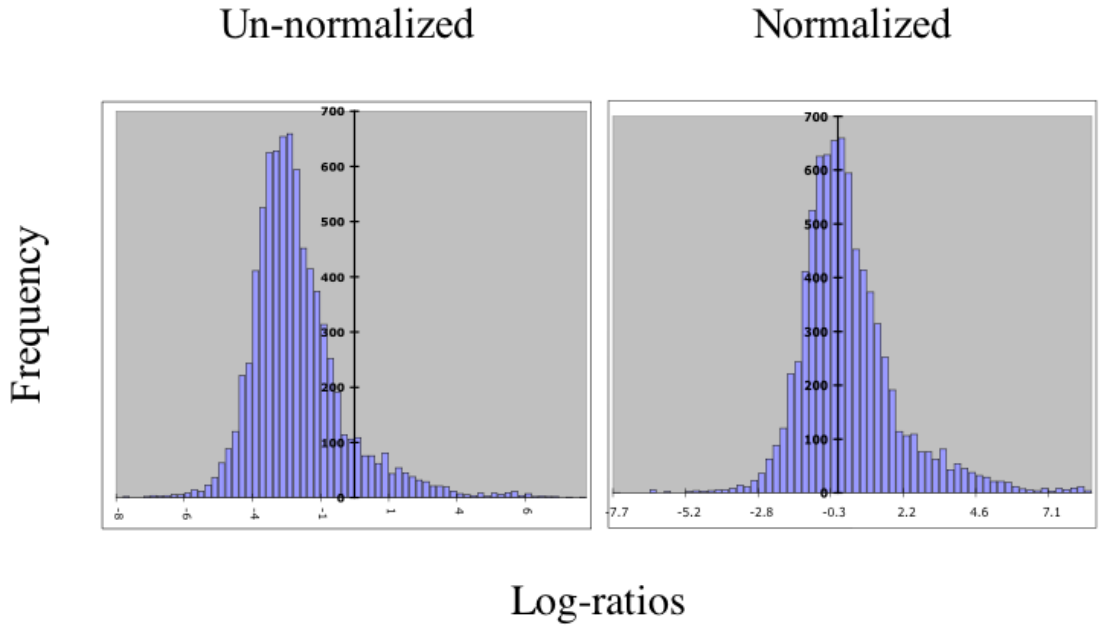
MPG miles per gallon
Cylinders number of cylinders
Displacement Engine displacement
Horsepower horsepower of the model
Weight how heavy is this model
Acceleration acceleration ability
Model Year which year this model is built
Origin either Europe, Japan or USA

housing.data

CRIM per capita crime rate by town
ZN proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS proportion of non-retail business acres per town
CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX nitric oxides concentration (parts per 10 million)
RM average number of rooms per dwelling
AGE proportion of owner-occupied units built prior to 1940
DIS weighted distances to five Boston employment centres
RAD index of accessibility to radial highways
TAX full-value property-tax rate per 10,000usd
PTRATIO pupil-teacher ratio by town
B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT % lower status of the population
PRICE price of property

Feature Selection





preprocessing.StandardScaler()

preprocessing.Normalization()

Keras Sequential Model

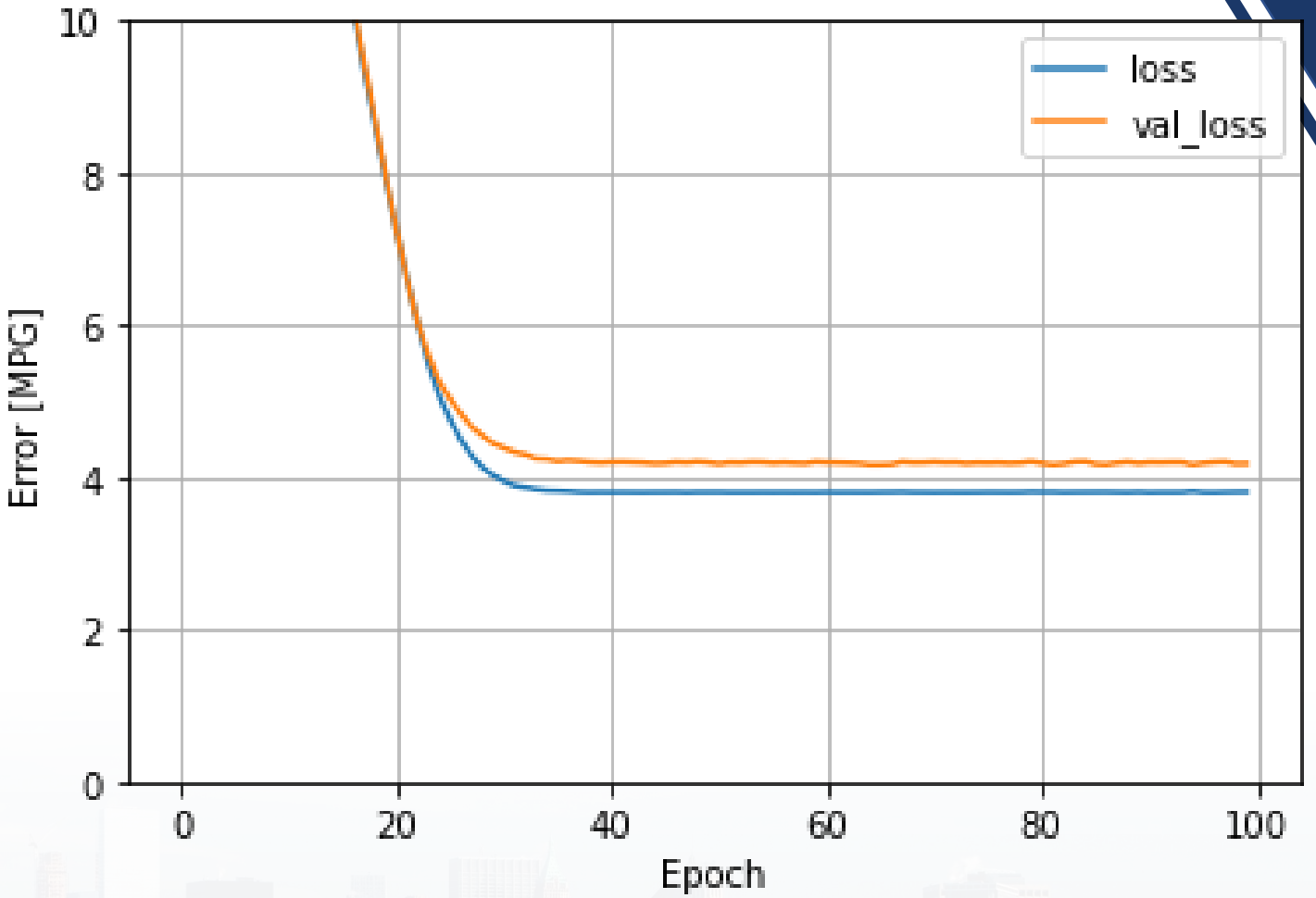
```
def model():
    # create model
    model = Sequential()
    model.add(normalization)
    model.add(Dense(64, activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    # Compile model
    model.compile(loss='mean_absolute_error',
optimizer=tf.optimizers.Adam(learning_rate=0.1))
    return model

model.summary()
```

How to fit

```
model.fit(train_features, train_labels, epochs=100,
verbose=0, validation_split = 0.2)
```

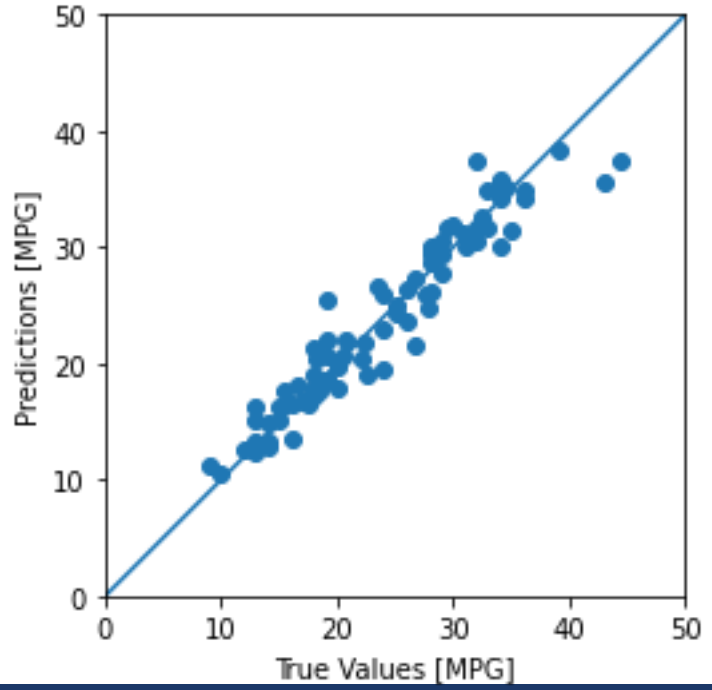
Accuracy

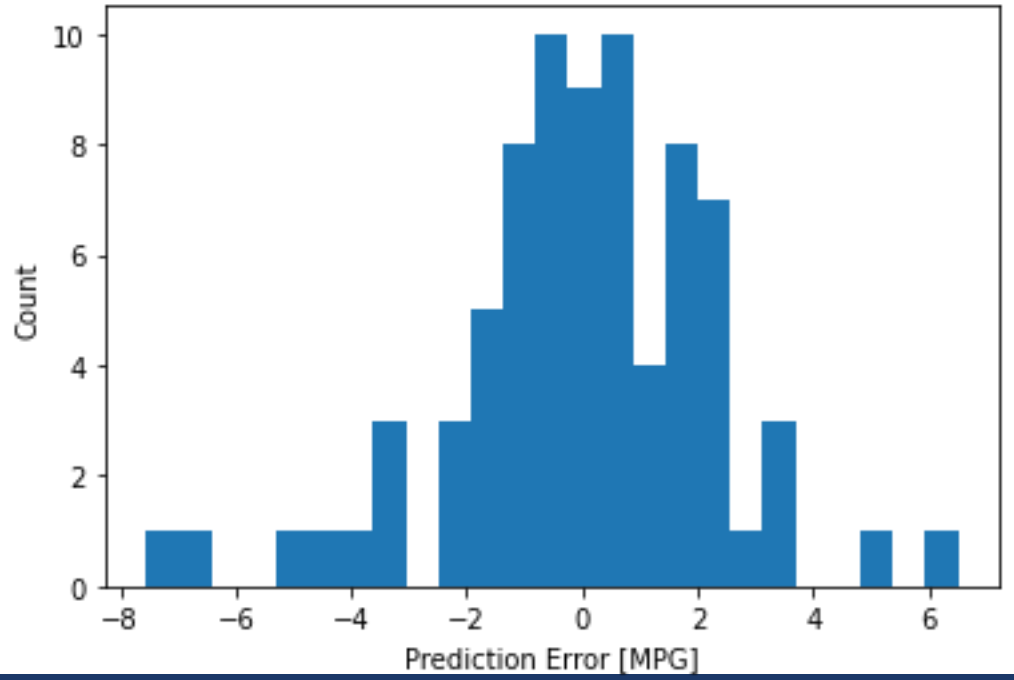
	loss	val_loss	epoch
95	3.801957	4.197738	95
96	3.804419	3.206789	96
97	3.802792	4.187988	97
98	3.803811	4.175385	98
99	3.804383	4.177610	99



Finally, predict new values

```
# to save a model for future use
dnn_model.save('dnn_model')
# to load a model
dnn_model = tf.keras.models.load_model('dnn_model')
# predict
test_predictions = dnn_model.predict(test_features).flatten()
```





Exercise

auto-mpg.data

DNN Model Loss (MAE) < 3

housing.data

DNN Model Loss (MSE) < 25

End of Lab 13