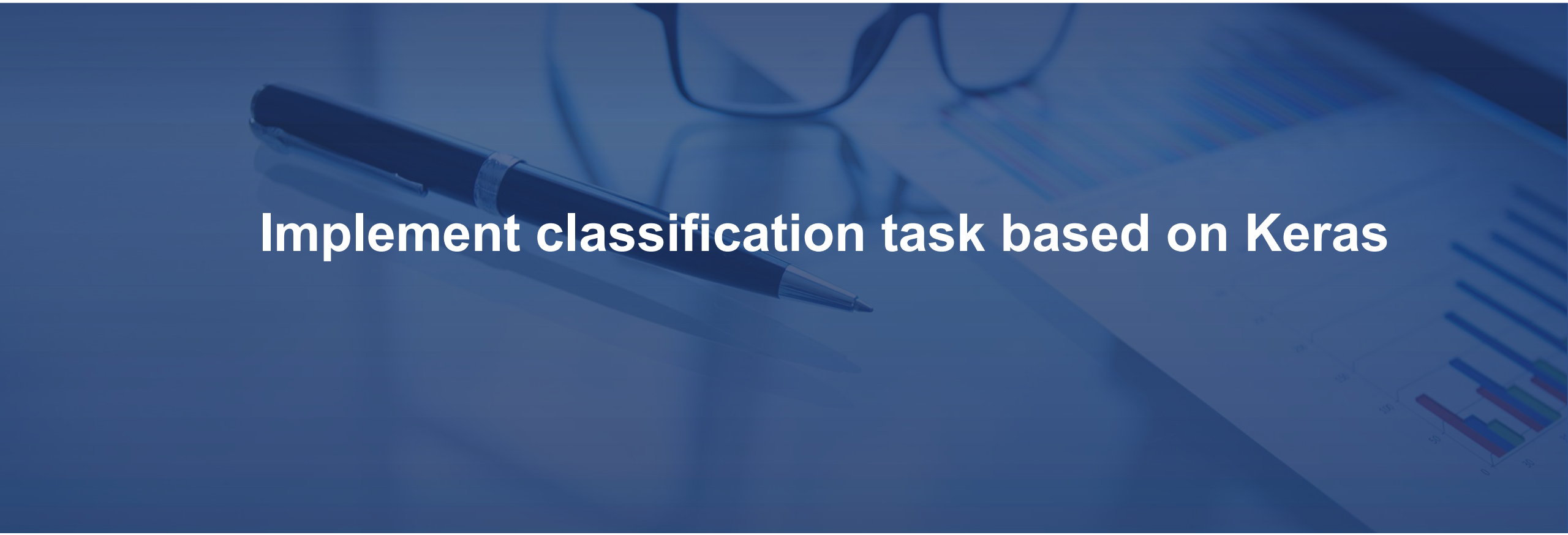
Knowledge Discovery and Data Mining

Lab 14 Using Keras 2 Classification Task

Xuan Song
Songx@sustech.edu.cn

Topics



Implement classification task based on Keras

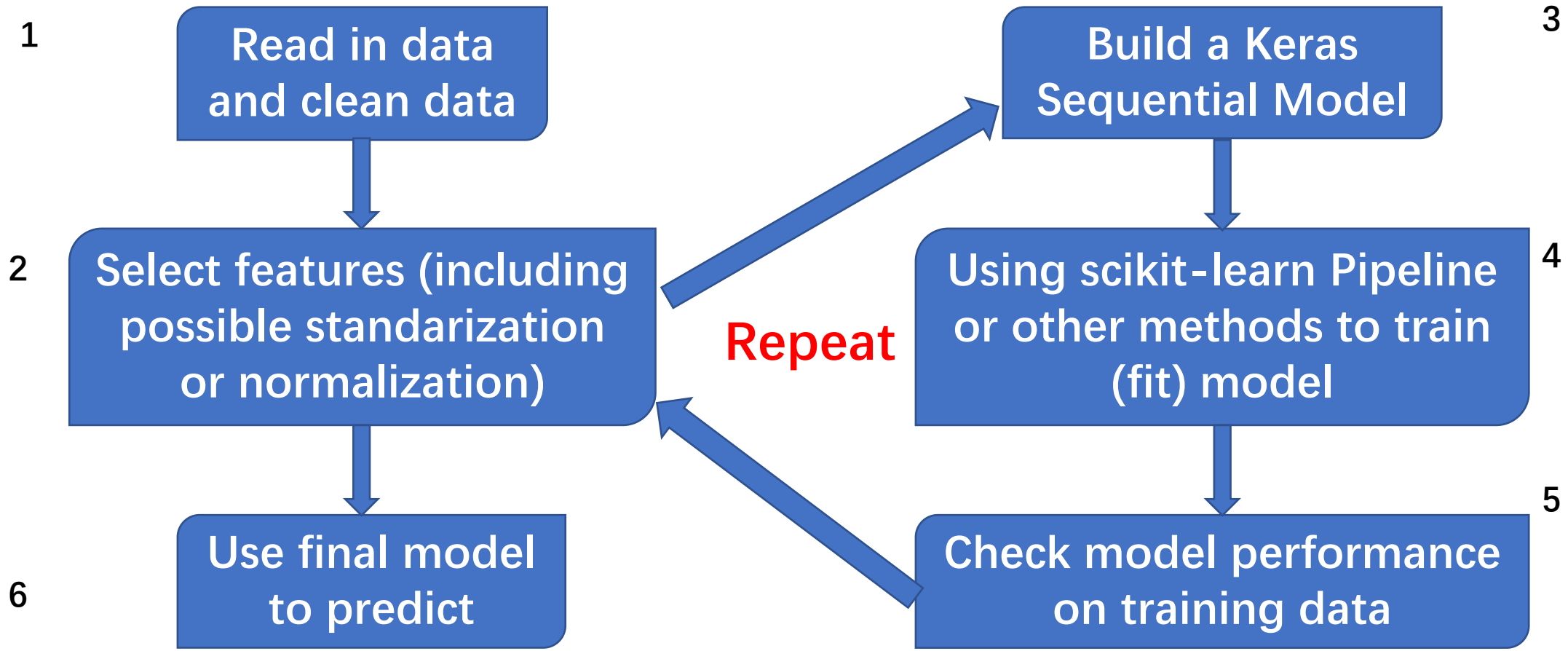
Implementing classification task based on Keras

- Sample data: diabetes.csv

1. Number of times pregnant
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

diabetes.csv

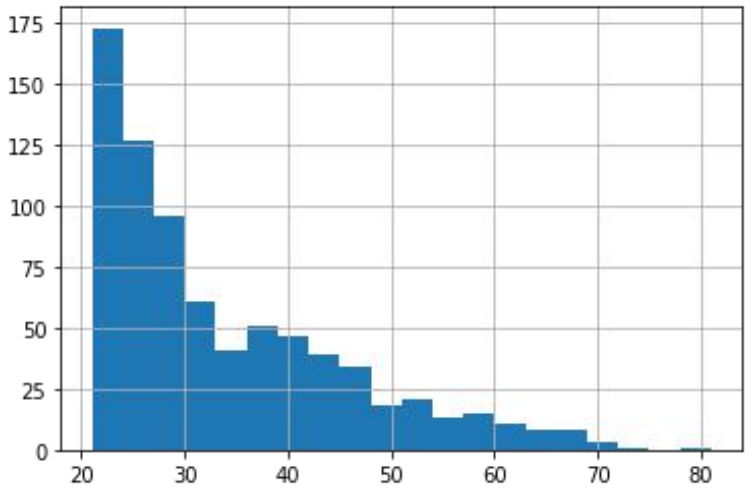
Implementing classification task based on Keras



Implementing classification task based on Keras

1. Load data from csv files.
2. Data preprocessing:
   (1) Normalization or standardization.
   (2) Encode age to age groups.

```
diabetes['Age'].hist(bins=20)

bins = [0,30,50,70,100]
labels =[0,1,2,3]
diabetes["Age_groups"] = pd.cut(diabetes["Age"],bins=bins,
labels=labels, include_lowest=True)
```



   (3) Label encoding

```
from tensorflow.python.keras.utils.np_utils import to_categorical
Y_train_binary= to_categorical(Y_train)
Y_test_binary = to_categorical(Y_test)
```

Implementing classification task based on Keras

3. Build a model.

```
def keras_model():
    #create a model
    model = Sequential()
    model.add(Dense(20, input_dim = X_train.shape[1], activation = 'relu'))
    model.add(Dense(10,activation = 'relu'))
    model.add(Dense(2, activation = 'softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer=tf.optimizers.Adam(learning_rate=0.01),
metrics=['accuracy'])

    return model

#Instantiate your model
model = keras_model()
```

https://keras.io/zh/losses/#categorical_crossentropy

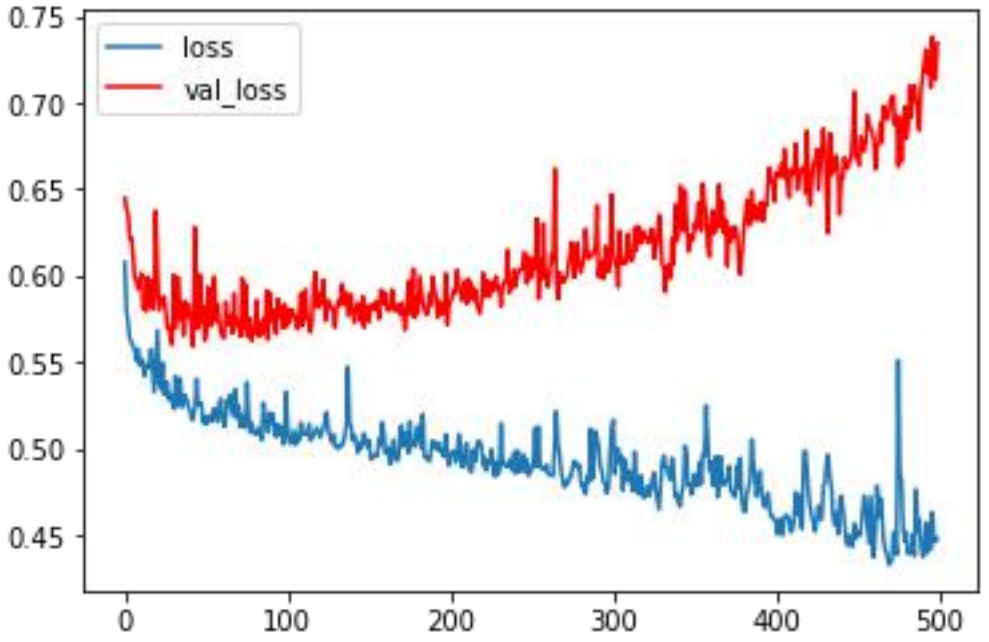
Implementing classification task based on Keras

4. Model fitting.

```
H = model.fit(X_train, Y_train_binary, validation_data=(X_test, Y_test_binary),epochs = 500, verbose =0)
```

5. Plot loss.

```
plt.plot(H.history["loss"],label="loss")
plt.plot(H.history["val_loss"], 'r',label="val_loss")
plt.legend()
```



Overfitting

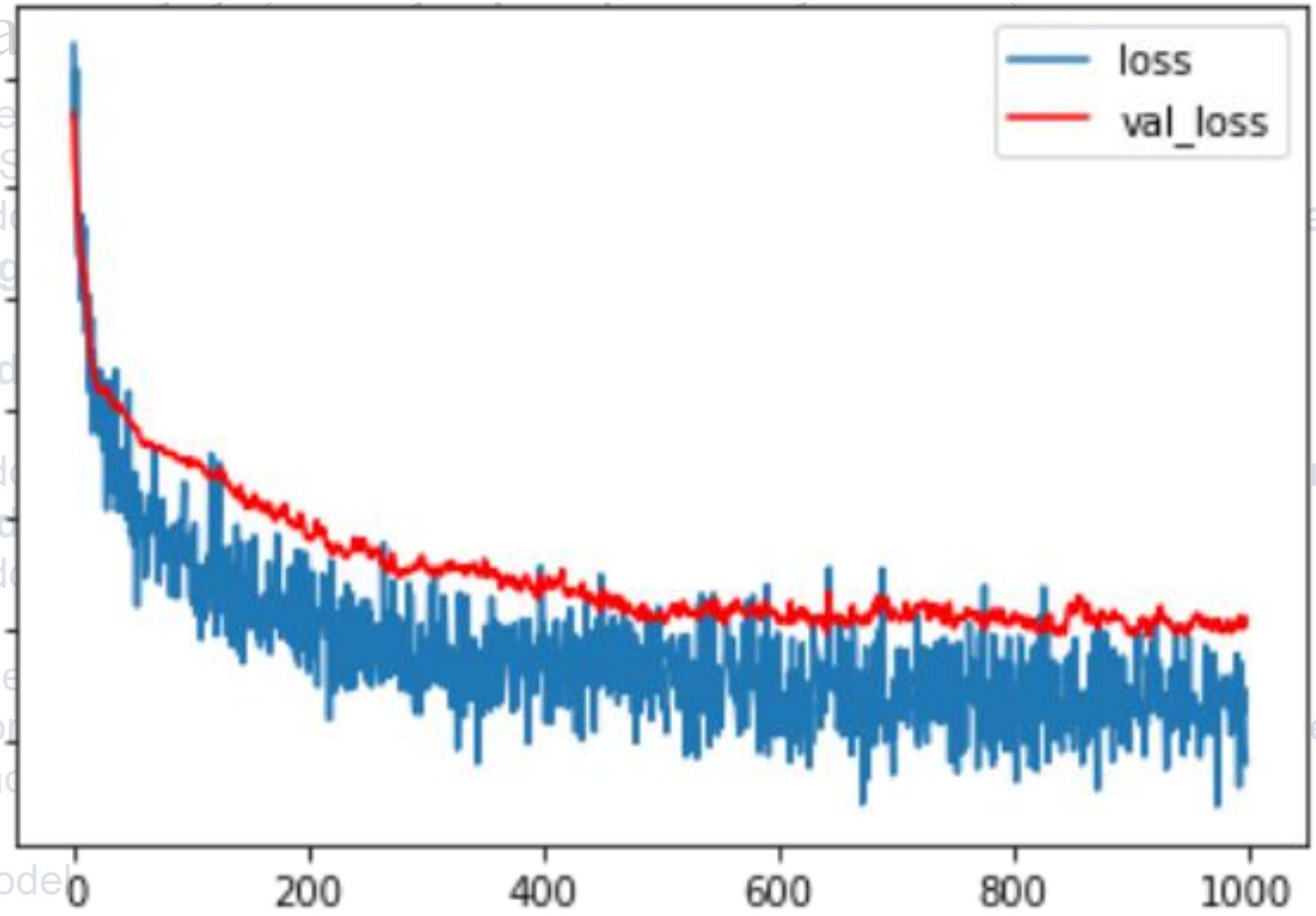
Implementing classification task based on Keras

3. Build a model (regularization + dropout).

```
def advanced_model():
    model = Sequential()
    model.add(Dense(20,input_dim = X_train.shape[1],activation = 'relu', kernel_regularizer =
    keras.regularizers.l2(0.001)))

    model.add(keras.layers.Dropout(0.5))

    model.add(Dense(10,activation = 'relu',kernel_regularizer = keras.regularizers.l2(0.001)))
    model.add(keras.layers.Dropout(0.5))
    model.add(Dense(2, activation = 'softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer=tf.optimizers.Adam(learning_rate=0.001),
 metrics=['accuracy'])

    return model

model = advanced_model()
```

Implementing classification task based on Keras

3. Build a




```
def advance
    model = S
    model.ad                                                    egularizer =
    keras.reg
    model.ad                                                    rs.l2(0.001)))
    model.ad
    model.ad
# Compile
    model.co                                                    earning_rate=0.001),
metrics=['ac
    return model

model = advanced_model()
```

Implementing classification task based on Keras

6. Prediction and evaluation.

```
import numpy as np
from sklearn.metrics import classification_report

y_pred_softmax = model.predict(X_test)
y_pred = np.argmax(y_pred_softmax, axis=1)
print(classification_report(Y_test,y_pred))
```

Task

- Implementing classification task based on Keras.

Dataset :Iris Data Set

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Class:

Iris Setosa
Iris Versicolour
Iris Virginica

iris.csv

End of Lab14