# Examples of code for data transfer, neural network classification, and using transfer learning

* **Transferring data from a CSV file for preprocessing**

Import Pandas as PD and Numpy as NP from Sklearn. preprocessing import StandardScaler

# Load data from a CSV file
data = pd.read\_csv('health\_data.csv')

# Preprocess the data
X = data.drop(['diagnosis'], axis = 1).
y = data['diagnosis']
scaler = StandardScaler()
X\_scaled = scaler.fit\_transform(X)

* **Developing a simple neural network classification model**

import tensorflow as tf

# Define the model architecture
model = tf.keras.models.Sequential([
   tf.keras.layers. Dense (64, activation='relu', input\_shape=(30,)),
   tf.keras.layers Dense (64, activation='relu'),
   tf.keras.layers. Dense(1, activation='sigmoid')
])

# Compile the model
model.compile (optimizer='Adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X\_scaled, y, epochs=10, batch\_size=32, validation\_split=0.2)

* **Transfer learning is used to refine a previously-trained image classification model**

import tensorflow as tf
from tensorflow.keras.applications import VGG16

# Load the pre-trained VGG16 model
base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3);

# Freeze the layers in the base model
for layer in base\_model.layers:
    layer.trainable = False

# Add new layers on top of the base model
x = base\_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(128, activation='relu')(x)
predictions = tf.keras.layers.Dense(2, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base\_model.input, outputs=predictions)

# Compile the model
model.compile (optimizer='Adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(X\_val, y\_val))

###

### **Citations**

### Transferring data from a CSV file for preprocessing

* Pandas: McKinney, Wes "Data structures for statistical computing in Python." Proceedings of the 9th Python in Science Conference, 2010.
* NumPy: van der Walt, Stefan, et al. "The NumPy array: a structure for efficient numerical computation." Computing in Science and Engineering 13.2 (2011): 22–30.
* Scikit-Learn: Pedregosa, Fabian, et al. "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825–2830.

### Developing a simple neural network classification model

* TensorFlow: Abadi, Martn, et al. "TensorFlow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.
* Keras: Chollet, François "Keras: The Python Deep Learning Library." ASC:1806.022, Astrophysics Source Code Library, 2018.

### Transfer learning is used to refine a previously-trained image classification model.

* TensorFlow: Abadi, Martn, et al. "TensorFlow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.
* Keras: Chollet, François "Keras: The Python Deep Learning Library." ASC:1806.022, Astrophysics Source Code Library, 2018.
* VGG16: Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognitionnal networks for large-scale image recognition" arXiv preprint arXiv:1409.1556 (2014)