National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Sharath Kasaraghatta Thimmaraya Gowda
Student ID: x20117507

School of Computing
National College of Ireland

Supervisor: Dr. Rashmi Gupta

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sharath Kasaraghatta Thimmaraya Gowda
Student ID:	x20117507
Programme:	Programme Name
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Rashmi Gupta
Submission Due Date:	16th August 2021
Project Title:	Configuration Manual
Word Count:	681
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sharath Kasaraghatta Thimmaraya Gowda
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sharath Kasaraghatta Thimmaraya Gowda
x20117507

1 Introduction

The Configuration manual is a guidance report which gives details of the step by step guide for project with respect to development, installation, implementation and deployment of the project 'Machine Learning Applications in Predicting Breast Cancer Survival Using Gene Information' presented in technical report.The purpose of this report is to support and guide through each stage in order to achieve the desired output and results, which are provided in the technical report. The complete project is built with a variety of technologies, libraries, hardware, and software combinations.

1.1 Project Overview

The project has two aspects with it, one predict breast cancer at early stage and second predict the survivability of breast cancer patients with gene information. There are several experiments conducted respect to these two health datasets,but the methods mainly made use in the project are Multiple instance learning and Tensorflow boosted tree estimators respectively for their health datasets. Both the algorithms yielded good results.

2 Pre-requisites

The following are the prerequisites: The software and hardware configurations are presented below. To train the model for such a large image datasets, the GPU (Graphics Processing Unit) is required.

2.1 Hardware Requirements

- Processor Required: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50GHz
- RAM: 8GB
- System Type: 64 bit Operating Systems
- ROM: 1TB HDD
- Operating System: Windows 10

2.2 Software Requirements

1. Web Browser: The Google Chrome web browser version 87.0.4280.88 was used to complete this project. Earlier versions of Google Chrome, on the other hand, do support colab notebook.

2. Google Colaboratory (colab) IDE: This project is built with Google Colaboratory (colab) IDE, a free cloud service that comes with numerous Artificial Intelligence (AI) libraries and a powerful GPU.

3. Google Colaboratory (colab) IDE: This project is built with Google Colaboratory (colab) IDE, a free cloud service that comes with numerous Artificial Intelligence (AI) libraries and a powerful GPU.

4. Python Libraries

▾ Libraries

```
[2] import fnmatch
    from glob import glob
    import random
    # from turtle import pd

    import cv2
    from flatbuffers.builder import np
    from keras.utils.np_utils import to_categorical
    from networkx.drawing.tests.test_pylab import plt
    from sklearn.model_selection import train_test_split
    import seaborn as sns
    # from sns import sns
```

```
[3] # CNN algo
    import matplotlib
    import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)from numba import njit

    #VISUALIZATION
    #from matplotlib.colors import ListedColormap
    import matplotlib.pyplot as plt
    #from pasta.augment import inline
    from scipy.constants import mil
    from tensorboard.notebook import display

    import seaborn as sns
    sns.set()
    import cv2 as cv

    #MACHINELEARNING
    #from sklearn import svm
    from sklearn.model_selection import train_test_split, StratifiedKFold
    from sklearn.decomposition import PCA

    from glob import glob
    from skimage import io
    from os import listdir
    import pickle

    import time
    import copy
    from tqdm.notebook import tqdm
```

Figure 1: Python Libraries for Dataset 1(Breast Cancer Prediction using Histopathological Images)

```
#Basic libraries
import numpy as np
import pandas as pd
#from scipy import stats

# Visualization libraries
import matplotlib.pyplot as plt
#import matplotlib
import seaborn as sns
#import yellowbrick as yb
from matplotlib.colors import ListedColormap
#from yellowbrick.classifier import ROCAUC
from matplotlib_venn import venn3
#import matplotlib.patches as mpatches

# Statistics, EDA, metrics libraries
#from scipy.stats import normaltest, skew
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler, MinMaxScaler
from sklearn.metrics import r2_score, median_absolute_error, mean_absolute_error, accuracy_score, f1_score
from sklearn.metrics import median_absolute_error, mean_squared_error, mean_squared_log_error
from sklearn.metrics import classification_report, confusion_matrix
#from scipy.special import boxcox, inv_boxcox

# Modeling libraries
#from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, StratifiedKFold, cross_val_predict,  KFold
from sklearn.neighbors import KNeighborsClassifier
#from sklearn.linear_model import LogisticRegression
#from sklearn.tree import DecisionTreeClassifier
#from sklearn.ensemble import RandomForestClassifier
#from sklearn.ensemble import ExtraTreesClassifier
#from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import roc_curve, auc, classification_report
from sklearn.decomposition import PCA
from scipy.stats import zscore
from itertools import combinations
from xgboost.sklearn import XGBClassifier
#from sklearn.ensemble import IsolationForest
#from sklearn.cluster import KMeans
import tensorflow as tf
```

Figure 2: Libraries for Dataset 2(Breast Cancer Survival Prediction using Gene Information)

3 Data Collection

The Datasets for this project is collected from Kaggle. Kaggle API is used to download the files into colab. Steps for collecting the data are below:

- First the key from Kaggle is copied and stored in google drive Figure 3

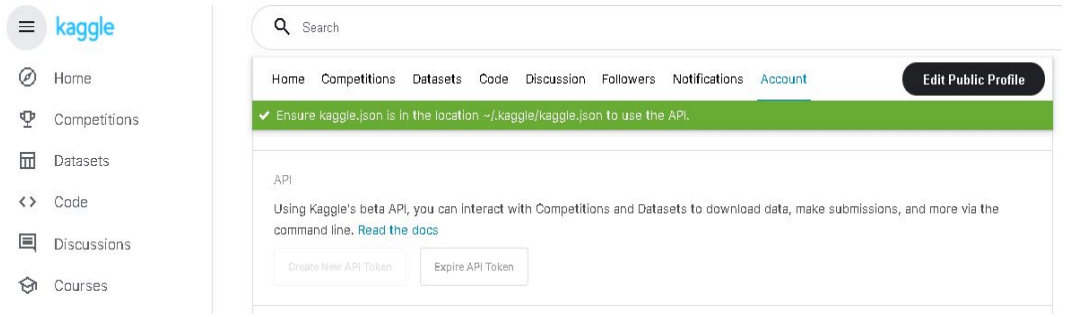


Figure 3: Kaggle API

- The Google Drive is then mounted onto Google Colab Figure 4

  The dataset used for Breast Cancer prediction using Histopathological images is collected by Janowczyk and Madabhushi (2016) and the dataset to predict Breast Cancer Survivability is collected by Pereira et al. (2016).

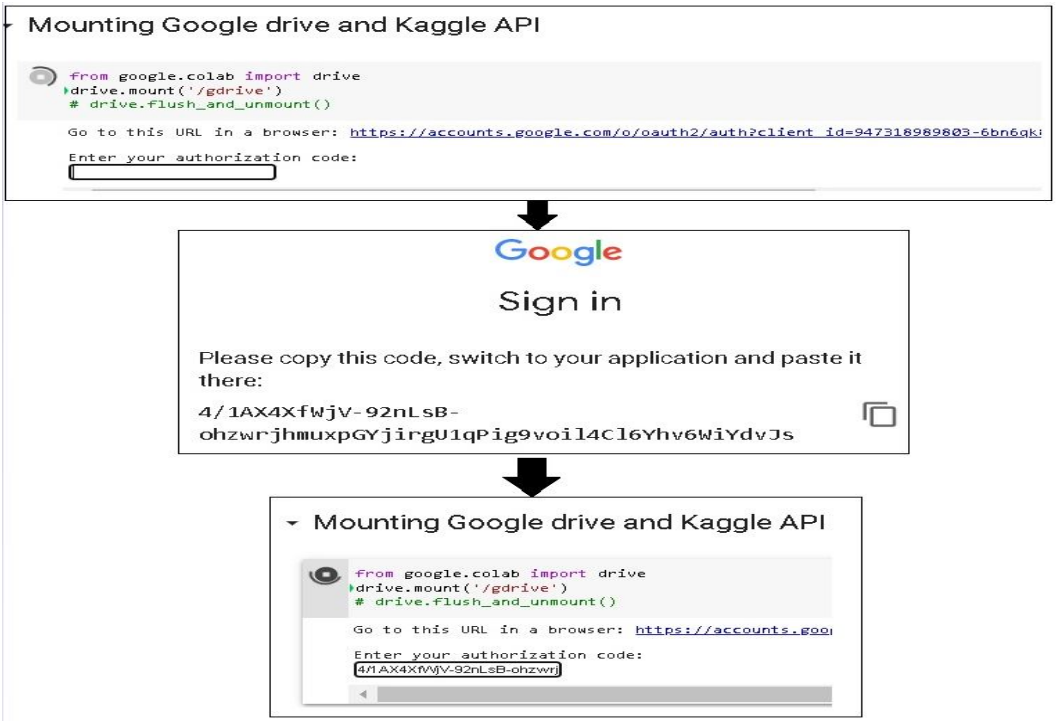


Figure 4: Mounting Google Drive onto Google Colab

- Request the Kaggle dataset required for project into the Google Colab virtual folders, then download and unzip the folder for further programming Figure 5

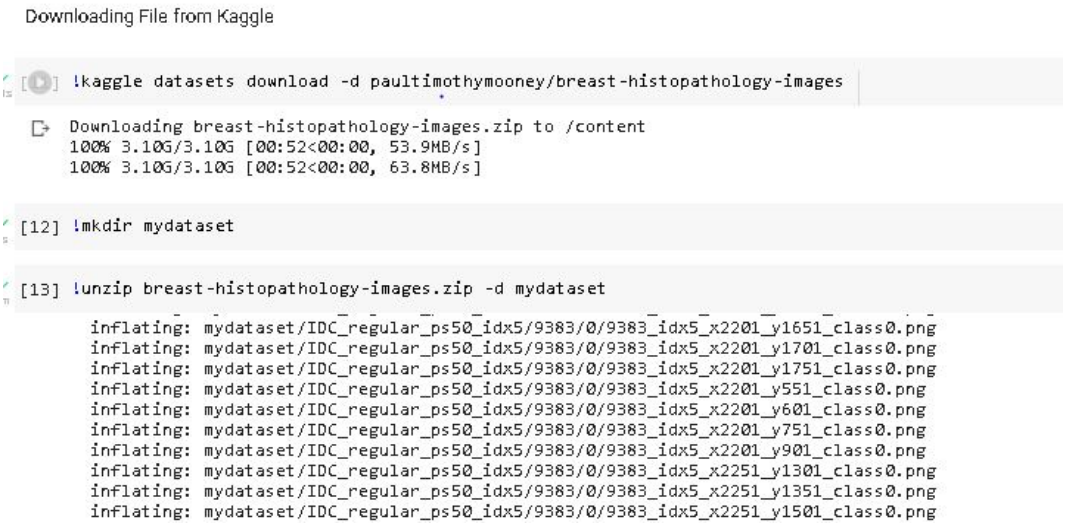


Figure 5: Downloading dataset from Kaggle

4 Data Preprocessing

Figure6,7 shows the pre-preprossing steps involved in dataset 1.

Handelling Class imbalances using random overSampler and underSampler

```
[ ] from imblearn.over_sampling import RandomOverSampler
    from imblearn.under_sampling import RandomUnderSampler
    #ros = RandomOverSampler(ratio='auto')
    ros = RandomUnderSampler(ratio='auto')
    X_trainRos, Y_trainRos = ros.fit_sample(X_trainFlat, Y_train)
    X_testRos, Y_testRos = ros.fit_sample(X_testFlat, Y_test)

    # Encode labels to hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
    Y_trainRosHot = to_categorical(Y_trainRos, num_classes = 2)
    Y_testRosHot = to_categorical(Y_testRos, num_classes = 2)
    #print("X_train: ", X_train.shape)
    #print("X_trainFlat: ", X_trainFlat.shape)
    #print("X_trainRos Shape: ",X_trainRos.shape)
    #print("X_testRos Shape: ",X_testRos.shape)
    #print("Y_trainRosHot Shape: ",Y_trainRosHot.shape)
    #print("Y_testRosHot Shape: ",Y_testRosHot.shape)

    for i in range(len(X_trainRos)):
        height, width, channels = 50,50,3
        X_trainRosReshaped = X_trainRos.reshape(len(X_trainRos),height,width,channels)
    #print("X_trainRos Shape: ",X_trainRos.shape)
    #print("X_trainRosReshaped Shape: ",X_trainRosReshaped.shape)

    for i in range(len(X_testRos)):
        height, width, channels = 50,50,3
        X_testRosReshaped = X_testRos.reshape(len(X_testRos),height,width,channels)
    #print("X_testRos Shape: ",X_testRos.shape)
    #print("X_testRosReshaped Shape: ",X_testRosReshaped.shape)

    dfRos = pd.DataFrame()
    dfRos["labels"]=Y_trainRos
    labRos = dfRos['labels']
    distRos = lab.value_counts()
    sns.countplot(labRos)
    print(dict_characters)
```

Figure 6: Handling Class Imbalance using Random UnderSampler and OverSampler

```
[25] def describeData(a,b):
         print('Total number of images: {}'.format(len(a)))
         print('Number of IDC(-) Images: {}'.format(np.sum(b==0)))
         print('Number of IDC(+) Images: {}'.format(np.sum(b==1)))
         print('Percentage of positive images: {:.2f}%'.format(100*np.mean(b)))
         print('Image shape (Width, Height, Channels): {}'.format(a[0].shape))
     describeData(X2,Y2)

     Total number of images: 10000
     Number of IDC(-) Images: 7403
     Number of IDC(+) Images: 2597
     Percentage of positive images: 25.97%
     Image shape (Width, Height, Channels): (50, 50, 3)
```

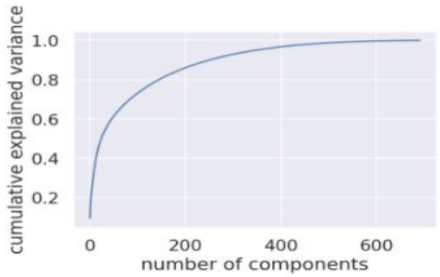
```
     dict_characters = {0: 'IDC(-)', 1: 'IDC(+)'}
     print(df.head(10))
     print("")
     print(dict_characters)
     def plotOne(a,b):
         """
         Plot one numpy array
         """
         plt.subplot(1,2,1)
         plt.title('IDC (-)')
         plt.imshow(a[0])
         plt.subplot(1,2,2)
         plt.title('IDC (+)')
         plt.imshow(b[0])
     plotOne(imgs0, imgs1)
```

```
                                                images  labels
0  [[[177, 140, 183], [174, 144, 187], [164, 129,...       1
1  [[[183, 154, 200], [158, 118, 164], [137, 93, ...       1
2  [[[182, 152, 196], [191, 159, 198], [194, 162,...       1
3  [[[180, 147, 201], [189, 163, 206], [178, 140,...       1
4  [[[175, 139, 177], [195, 161, 195], [191, 167,...       1
5  [[[190, 159, 199], [193, 160, 204], [181, 147,...       1
6  [[[169, 131, 166], [166, 125, 166], [195, 171,...       1
7  [[[163, 119, 160], [175, 132, 172], [180, 149,...       1
8  [[[193, 172, 214], [175, 142, 187], [162, 122,...       1
9  [[[169, 135, 182], [171, 133, 178], [199, 177,...       1

{0: 'IDC(-)', 1: 'IDC(+)'}
```

Figure 7: Seprating data into IDC+ and IDC-

```
[ ] tmppca = PCA().fit(processedMinMaxDf)
    plt.plot(np.cumsum(tmppca.explained_variance_ratio_))
    plt.xlabel('number of components')
    plt.ylabel('cumulative explained variance');
```




```
[ ] pca = PCA(n_components=100)
    principalComponents = pca.fit_transform(processedMinMaxDf)
    principalDf = pd.DataFrame(data = principalComponents)
    print(principalDf)
    processedMinMaxPrincipalDf = principalDf
```

```
             0         1         2   ...        97        98        99
0    -0.800953 -0.307617  1.153655  ... -0.245867  0.224169 -0.243908
1     0.089689  0.140927  1.884968  ... -0.152081 -0.086253 -0.087221
2    -0.415908 -0.337168  0.945563  ... -0.029858  0.136505  0.149838
3    -0.624155  0.936649  1.346201  ...  0.033430  0.212779 -0.032253
4    -0.078409  1.081603  0.572703  ...  0.321027 -0.087593  0.011742
...        ...       ...       ...  ...       ...       ...       ...
1087 -1.212154 -0.197047 -0.178630  ...  0.126846 -0.126435  0.000311
1088 -0.913070 -0.200747 -0.810680  ...  0.107206 -0.070676  0.053326
1089  2.166044  0.431529 -0.243436  ... -0.080297  0.147676 -0.105401
1090 -0.655859 -0.086812 -0.694149  ... -0.147275  0.091186  0.032287
```

Figure 8: Principal Component Analysis

5 Model Implementation

This study revolves around treatment detection and treatment of Breast Cancer. This involves two primary problems:

- Detect presence of cancer, based on images of scans
- Predict the survivability of a patient diagnosed with cancer, based on various clinical and genetic information of the patient

Multiple classifiers are proposed for answering these two questions, out which Attention based Multiple Instance learning, Tensorflow boosted tree estimator,Convolutional Nueral Networks, Support vector Classifier, Multilayer perceptron are used in the project.

Attention based Multiple Instance Learning

Attention based Multiple instance Learning is used for predicting breast cancer using histopathological images.

+ Code + Text

```
[ ] import torch
    import torch.nn as nn
    import torch.nn.functional as F


    class Attention(nn.Module):
        def __init__(self):
            super(Attention, self).__init__()
            self.L = 500
            self.D = 128
            self.K = 1

            self.feature_extractor_part1 = nn.Sequential(
                # nn.Conv2d(1, 20, kernel_size=5),
                # nn.Flatten(),
                nn.Conv2d(3, 20, kernel_size=5),
                # nn.ReLU(),
                # nn.MaxPool2d(2, stride=2),
                # nn.Conv2d(20, 50, kernel_size=5),
                nn.ReLU(),
                nn.MaxPool2d(2, stride=2)
            )

            self.feature_extractor_part2 = nn.Sequential(
                # nn.Linear(50 * 4 * 4, self.L),
                nn.Linear(20 * 23 * 23, self.L),
                nn.ReLU(),
            )

            self.attention = nn.Sequential(
                nn.Linear(self.L, self.D),
                nn.Tanh(),
                nn.Linear(self.D, self.K)
            )

            self.classifier = nn.Sequential(
                nn.Linear(self.L*self.K, 1),
                nn.Sigmoid()
            )

        def forward(self, x):
            x = x.unsqueeze(0)
            # x = x.unsqueeze(0)
            H = self.feature_extractor_part1(x)
            # print("FEATURE EXTRACTOR - 1:", H.shape)
```

Figure 9: Building Attention model

```
[58] def runKerasCNNAugment(a,b,c,d,e,f):
        """
        Run Keras CNN: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py
        """
        batch_size = 128
        num_classes = 2
        epochs =80
        #     img_rows, img_cols = a.shape[1],a.shape[2]
        img_rows,img_cols=50,50
        input_shape = (img_rows, img_cols, 3)
        model = Sequential()
        model.add(Conv2D(32, kernel_size=(3, 3),
                         activation='relu',
                         input_shape=input_shape,strides=e))
        model.add(Conv2D(64, (3, 3), activation='relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Dropout(0.25))
        model.add(Flatten())
        model.add(Dense(128, activation='relu'))
        model.add(Dropout(0.5))
        model.add(Dense(num_classes, activation='softmax'))
        model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adadelta(),
                      metrics=['accuracy'])
        datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # divide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range=20,  # randomly rotate images in the range (degrees, 0 to 180)
            width_shift_range=0.2,  # randomly shift images horizontally (fraction of total width)
            height_shift_range=0.2,  # randomly shift images vertically (fraction of total height)
            horizontal_flip=True,  # randomly flip images
            vertical_flip=True)  # randomly flip images
        ##########################################################################################################
        # history = model.fit_generator(datagen.flow(a,b, batch_size=32),
        #                     steps_per_epoch=len(a) / 32, epochs=epochs,class_weight=f, validation_data = [c, d],callbacks = [MetricsCheckpoint('logs')])
        # history = model.fit(datagen.flow(a,b, batch_size=32),
        #                     steps_per_epoch=len(a) / 32, epochs=epochs,class_weight=f, validation_data = [c, d],callbacks = [MetricsCheckpoint('logs')])
        history = model.fit(a, b, steps_per_epoch=len(a) / 32,
                            epochs=epochs,callbacks = [MetricsCheckpoint('logs')])
        score = model.evaluate(c,d, verbose=0)
        print('\nKeras CNN #1C - accuracy:', score[1],'\n')
        y_pred = model.predict(c)
        map_characters = {0: 'IDC(-)', 1: 'IDC(+)'}
```

Figure 10: Convolutional Nueral models

Convolutional Nueral Network

Convoultional Nueral Network is used for predicting Breast Cancer using Histopathological Images.

Support Vector Classifier, Multilayer Perceptron and K-Nearest Neighbours

These models are used to predict the survivability of breast cancer.

```
[ ]  def evaluateModel(classifier, classifierName, xTrain, yTrain, xTest, yTest, kfold=10):
         classifier.fit(xTrain, yTrain)
         results = cross_val_score(classifier, xTrain, yTrain, cv = kfold)
         print(classifierName)
         print("-"*50)
         print("CV scores: ", results)
         print("CV Standard Deviation: ", results.std())
         print()
         print('CV Mean score: ', results.mean());
         print('Train score:    ', classifier.score(xTrain, yTrain))
         print('Test score:     ', classifier.score(xTest, yTest))

         yPred = classifier.predict(xTest)
         report = classification_report(yTest, yPred)
         print("CLASSIFICATION REPORT:")
         print(report)

         prob = classifier.predict_proba(xTest)
         prob = prob[:, 1]
         # print(prob)
         # print(yTest)
         fpr, tpr, _ = roc_curve(yTest, prob)
         plt.plot(fpr, tpr)
         plt.show()

         print("="*100, "\n\n")
```

Figure 11: SVC,MLP and KNN models

Tensorflow Boosted tree Estimators
Tesorflow boosted tree Estimators is used to predict survivability of Breast Cancer.

```
[ ] # Use entire batch since this is a small dataset.
    NUM_EXAMPLES = len(y_train)

    def make_input_fn(X, y, n_epochs=None, shuffle=True):
      def input_fn():
        dataset = tf.data.Dataset.from_tensor_slices((dict(X), y))
        if shuffle:
          dataset = dataset.shuffle(NUM_EXAMPLES)
        # For training, cycle thru dataset as many times as need (n_epochs=None).
        dataset = dataset.repeat(n_epochs)
        # In memory training doesn't use batching.
        dataset = dataset.batch(NUM_EXAMPLES)
        return dataset
      return input_fn

    # Training and evaluation input functions.
    # print(dftrain)
    # print(y_train)



    train_input_fn = make_input_fn(dftrain, y_train)
    eval_input_fn = make_input_fn(dfeval, y_eval, shuffle=False, n_epochs=1)
```

```
    # Since data fits into memory, use entire dataset per layer. It will be faster.
    # Above one batch is defined as the entire dataset.
    n_batches = 1
    est = tf.estimator.BoostedTreesClassifier(feature_columns,
                                              n_batches_per_layer=n_batches)#, n_trees=100, max_depth=6)

    # The model will stop training once the specified number of trees is built, not
    # based on the number of steps.
    est.train(train_input_fn)#, max_steps=1)

    # Eval.
    result = est.evaluate(eval_input_fn)
    # clear_output()
    print(pd.Series(result))
```

Figure 12: Tensorflow Boosted tree Estimators

References

Janowczyk, A. and Madabhushi, A. (2016). Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases.
URL: *https://pubmed.ncbi.nlm.nih.gov/27563488/*

Pereira, B., Chin, S.-F., Rueda, O. M., Vollan, H.-K. M., Provenzano, E., Bardwell, H. A., Pugh, M., Jones, L., Russell, R., Sammut, S.-J. and et al. (2016). The somatic mutation profiles of 2,433 breast cancers refine their genomic and transcriptomic landscapes.
URL: *https://www.nature.com/articles/ncomms11479*