Exercise 1

(5 points) What is a perceptron? Be specific.

It is a single-layer neural network used in classification tasks while working with a set of input data. Since perceptron uses classified data points which are already labeled, it is a supervised learning algorithm. This algorithm is used to enable neurons to learn and process elements in the training set one at a time.

Exercise 2

(5 points) What are the different types of perceptrons? Briefly describe each of them.

There are two types of perceptrons:

- **Single-layer perceptrons:** single-layer perceptrons can learn only linearly separable patterns.
- **Multi-layer perceptrons:** multi-layer perceptrons, also known as feed-forward networks, contain at least one hidden layer with a non-linear activation function.

Exercise 3

(5 points) What is a hard margin in a support vector machine model? Be specific.

A hard-margin in a support vector machine model refers to fitting a model with zero errors in the training dataset.

Exercise 4

(4 points) The effectiveness of a support vector machine model depends on:

(a) kernel

(b) kernel parameters

(c) penalty cost parameter

(d) **All of the above**

(e) None of the above

Exercise 5

(4 points) What is/are **true** about kernel in SVM?

(a) Kernel function map low dimensional data into high dimensional space.

(b) Kernel function map high dimensional data into low dimensional space.

(c) It is a similarity function.

(d) **(a) and (c)**

(e) (b) and (c)

(f) (a), (b) and (c)

(g) None of the above

Exercise 6

Consider the `framingham.csv` data file. The dataset is publically available on the Kaggle website, and it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD). The dataset provides the patients? information. It includes over 4,000 records and 15 attributes. Each attribute is a potential risk factor. There are both demographic, behavioral and medical risk factors.

- Demographic:
  - Sex: male or female (Nominal)
  - Age: Age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)
- Behavioral
  - Current Smoker: whether or not the patient is a current smoker (Nominal)
  - Cigs Per Day: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)
- Medical (history)
  - BP Meds: whether or not the patient was on blood pressure medication (Nominal)
  - Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)
  - Prevalent Hyp: whether or not the patient was hypertensive (Nominal)
  - Diabetes: whether or not the patient had diabetes (Nominal)
- Medical (current)

  - Tot Chol: total cholesterol level (Continuous)
  - Sys BP: systolic blood pressure (Continuous)
  - Dia BP: diastolic blood pressure (Continuous)
  - BMI: Body Mass Index (Continuous)
  - Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)
  - Glucose: glucose level (Continuous)

- Predict variable (desired target)
  - 10 year risk of coronary heart disease CHD (binary: "1", means "Yes", "0" means "No")

In Python, answer the following:

(a) (4 points) Load the data file to you S3 bucket. Using the pandas library, read the csv data file and create a data-frame called `heart`.

```
import boto3
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, recall_score
from sklearn.svm import SVC
import tensorflow as tf

## Defining the bucket
s3 = boto3.resource('s3')
bucket_name = 'data-445'
bucket = s3.Bucket(bucket_name)

## Defining the csv file
file_key = 'Fall_2021/In_Class_Assignments/framingham.csv'

bucket_object = bucket.Object(file_key)
file_object = bucket_object.get()
file_content_stream = file_object.get('Body')

## Reading the csv file
heart = pd.read_csv(file_content_stream)
```

(b) (3 points) Remove observations with missing values.

```
## Removing observations with NAs
heart = heart.dropna()
```

(c) (40 points) Using `age`, `currentSmoker`, `totChol`, `BMI`, and `heartRate` as the predictor variables, and `TenYearCHD` as the target variable, do the following:

(i) Split the data into train (80%) and test (20%).

```
## Defining the input and target variables
X = heart[['age', 'currentSmoker', 'totChol', 'BMI', 'heartRate']]
Y = heart['TenYearCHD']

## Splitting the data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)
```

(ii) Using `MinMaxScaler`, transform all the input variables in the train and test datasets to 0-1 scale.

```
## Transforming the input variables
scaler_train = MinMaxScaler(feature_range = (0, 1)).fit(X_train)
X_train = scaler_train.transform(X_train)

scaler_test = MinMaxScaler(feature_range = (0, 1)).fit(X_test)
X_test = scaler_train.transform(X_test)
```

- Build a multi-layer perceptron model with one single hidden layer with 4 neurons (hyperbolic tangent as the activation function) and softmax as the activation function for the output. Use the stochastic descent gradient as the method to estimate the weights (`optimizer = 'sgd'`) and `metrics = ['accuracy']`. Use `epochs = 100` and `batch_size = 500` to build the model. After that, use the model to predict on the test dataset. Using 15% as the cut-off value, report the recall of this model.

```
mlp1 = tf.keras.models.Sequential([
      tf.keras.layers.Dense(4, input_dim = 5, activation = 'tanh'),
      tf.keras.layers.Dense(2, activation = 'softmax')
])

mlp1.compile(optimizer = 'sgd', loss = 'categorical_crossentropy',
             metrics = ['accuracy'])
```

```
## Building the model
mlp1.fit(X_train, tf.keras.utils.to_categorical(Y_train, num_classes = 2),
         epochs = 100,
         batch_size = 500,
         validation_data = (X_test,
                            tf.keras.utils.to_categorical(Y_test,
                            num_classes = 2)),
         verbose = 0)

## Extracting predictions
mlp_pred1 = mlp1.predict(X_test)[:, 1]

## Changing likelihood to labels
mlp_pred1 = np.where(mlp_pred1 < 0.15, 0, 1)

## Computing the recall
recall_score(Y_test, mlp_pred1)
```

- Build a multi-layer perceptron model with one single hidden layer with 4 neurons (ReLU as the activation function) and softmax as the activation function for the output. Use the stochastic descent gradient as the method to estimate the weights (`optimizer = 'sgd'`) and `metrics = ['accuracy']`. Use `epochs = 100` and `batch_size = 500` to build the model. After that, use the model to predict on the test dataset. Using 15% as the cut-off value, report the recall of this model.

```
mlp2 = tf.keras.models.Sequential([
      tf.keras.layers.Dense(4, input_dim = 5, activation = 'relu'),
      tf.keras.layers.Dense(2, activation = 'softmax')
])

mlp2.compile(optimizer = 'sgd', loss = 'categorical_crossentropy',
             metrics = ['accuracy'])

## Building the model
mlp2.fit(X_train, tf.keras.utils.to_categorical(Y_train, num_classes = 2),
         epochs = 100,
         batch_size = 500,
         validation_data = (X_test,
                            tf.keras.utils.to_categorical(Y_test,
                            num_classes = 2)),
         verbose = 0)

## Extracting predictions
mlp_pred2 = mlp2.predict(X_test)[:, 1]
```

```
## Changing likelihood to labels
mlp_pred2 = np.where(mlp_pred2 < 0.15, 0, 1)

## Computing the recall
recall_score(Y_test, mlp_pred2)
```

- Build a support vector machine model using `rbf` as the kernel. After that, use the model to predict on the test dataset. Using 15% as the cut-off value, report the recall of this model.

```
## Building the svm model
svm_md1 = SVC(kernel = 'rbf', probability = True).fit(X_train, Y_train)

## Predicting on the test dataset
svm_pred1 = svm_md1.predict_proba(X_test)[:, 1]

## Changing likelihood to labels
svm_pred1 = np.where(svm_pred1 < 0.15, 0, 1)

## Computing the recall
recall_score(Y_test, svm_pred1)
```

- Build a support vector machine model using `poly` as the kernel. After that, use the model to predict on the test dataset. Using 15% as the cut-off value, report the recall of this model.

```
## Building the svm model
svm_md2 = SVC(kernel = 'poly', probability = True).fit(X_train, Y_train)

## Predicting on the test dataset
svm_pred2 = svm_md2.predict_proba(X_test)[:, 1]

## Changing likelihood to labels
svm_pred2 = np.where(svm_pred2 < 0.15, 0, 1)

## Computing the recall
recall_score(Y_test, svm_pred2)
```

(d) (20 points) Repeat part (c) 100 times. Create a visualization that shows the recall value for each of the models at each iteration. Also, report the average recall of each of the model for the 100 repetitions. What model would use to predict `TenYearCHD`?

```
## Defining lists to store results
md1_recall = list()
md2_recall = list()
md3_recall = list()
md4_recall = list()

for i in range(0, 100):

    ## Splitting the data
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2)

    ## Transforming the input variables
    scaler_train = MinMaxScaler(feature_range = (0, 1)).fit(X_train)
    X_train = scaler_train.transform(X_train)

    scaler_test = MinMaxScaler(feature_range = (0, 1)).fit(X_test)
    X_test = scaler_train.transform(X_test)

    ##############
    ## Model 1 ##
    ##############

    mlp1 = tf.keras.models.Sequential([
           tf.keras.layers.Dense(4, input_dim = 5, activation = 'tanh'),
           tf.keras.layers.Dense(2, activation = 'softmax')
    ])

    mlp1.compile(optimizer = 'sgd', loss = 'categorical_crossentropy',
                 metrics = ['accuracy'])

    mlp1.fit(X_train, tf.keras.utils.to_categorical(Y_train, num_classes = 2),
             epochs = 100,
             batch_size = 500,
             validation_data = (X_test,
                                tf.keras.utils.to_categorical(Y_test,
                                num_classes = 2)),
             verbose = 0)

    mlp_pred1 = mlp1.predict(X_test)[:, 1]
    mlp_pred1 = np.where(mlp_pred1 < 0.15, 0, 1)
    md1_recall.append(recall_score(Y_test, mlp_pred1))

    ##############
    ## Model 2 ##
    ##############
```

```
    mlp2 = tf.keras.models.Sequential([
           tf.keras.layers.Dense(4, input_dim = 5, activation = 'relu'),
           tf.keras.layers.Dense(2, activation = 'softmax')
    ])

    mlp2.compile(optimizer = 'sgd', loss = 'categorical_crossentropy',
                 metrics = ['accuracy'])

    mlp2.fit(X_train, tf.keras.utils.to_categorical(Y_train, num_classes = 2),
             epochs = 100,
             batch_size = 500,
             validation_data = (X_test,
                                tf.keras.utils.to_categorical(Y_test,
                                num_classes = 2)),
             verbose = 0)

    mlp_pred2 = mlp2.predict(X_test)[:, 1]
    mlp_pred2 = np.where(mlp_pred2 < 0.15, 0, 1)
    md2_recall.append(recall_score(Y_test, mlp_pred2))

    ##############
    ## Model 3 ##
    ##############

    svm_md1 = SVC(kernel = 'rbf', probability = True).fit(X_train, Y_train)

    svm_pred1 = svm_md1.predict_proba(X_test)[:, 1]
    svm_pred1 = np.where(svm_pred1 < 0.15, 0, 1)
    md3_recall.append(recall_score(Y_test, svm_pred1))

    ##############
    ## Model 4 ##
    ##############

    svm_md2 = SVC(kernel = 'poly', probability = True).fit(X_train, Y_train)

    svm_pred2 = svm_md2.predict_proba(X_test)[:, 1]
    svm_pred2 = np.where(svm_pred2 < 0.15, 0, 1)
    md4_recall.append(recall_score(Y_test, svm_pred2))

iterations = range(0, 100)

import matplotlib.pyplot as plt
fig = plt.figure(figsize = (14, 10))
```

```
plt.plot(iterations, md1_recall, marker = 'o', color = 'blue',
         label = 'Model 1')
plt.plot(iterations, md2_recall, marker = 'o', color = 'orange',
         label = 'Model 2')
plt.plot(iterations, md3_recall, marker = 'o', color = 'brown',
         label = 'Model 3')
plt.plot(iterations, md4_recall, marker = 'o', color = 'black',
         label = 'Model 4')

plt.xlabel('Number of Iterations')
plt.ylabel('Recall')
plt.legend()
plt.grid()
plt.show()
```

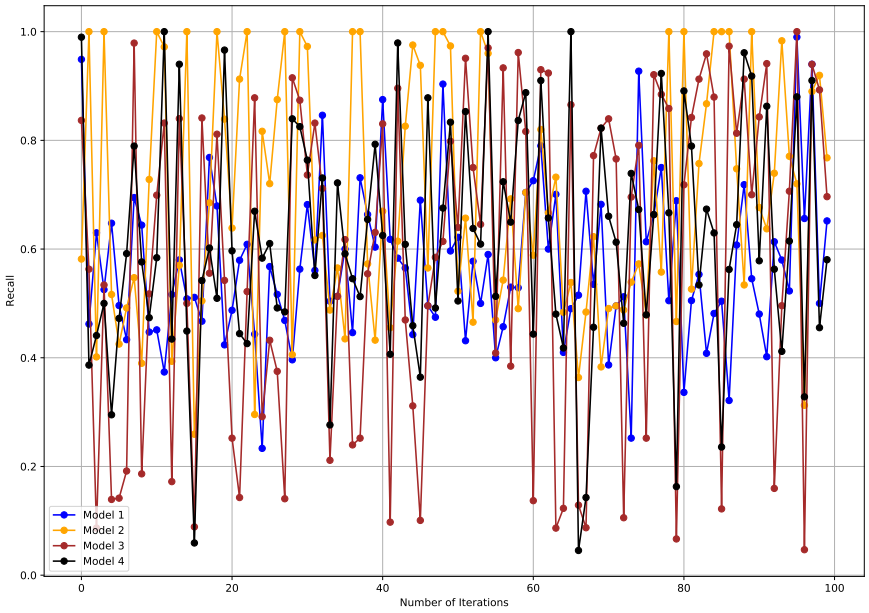

Figure 1: Recall of the four considered models

Based on my results, I would use the second model to predict `TenYearCHD`.